64-5--

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

AF-WP-O-OCT 62 3,500

PATTERN RECOGNITION WITH SELF-ORGANIZING MACHINES

THESIS

GE/EE/63-8

Alling C. Foreman
2/Lt USAF

AF-WP-O-OCT 63 24

PATTERN RECOGNITION WITH SELF-ORGANIZING MACHINES

THESIS

Presented to the Faculty of the School of Engineering of
the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

By
Alling C. Foreman, B.E.E.
2/Lt USAF

GE/EE/63-8
Graduate Electrical Engineering
August 1963

Preface

Pattern recognition using self-organizing machines was chosen as a thesis topic because of my rapidly increasing interest in electronic computers. The subject was suggested by Captain Frank M. Brown. He had found that no investigation and comparison of the pattern recognition capabilities of three classes of self-organizing machines existed in the literature.

It was first believed that a mathematical study in the form of theorems and proofs could be made. This idea was soon discarded because it was far too complex. A period of thrashing followed during which attempts were made to find a method of attacking the problem. It was finally decided to work through a large number of pattern recognition problems using mathematical models of typical machines from each of the three classes.

Because each problem required a very large number of simple calculations and comparisons, computer programs were written to simulate the machines. Mathematical models suitable for computer programming had to be derived from the somewhat confusing literature available. While the mathematical models were being developed, a more thorough understanding of some of the basic pattern recognition

techniques was acquired.

It soon became evident that the machines being studied were not (as Voltaire's Candide would say) the best possible machines in this best of all possible worlds. Therefore, an attempt was made to design a simpler and yet more capable self-organizing pattern recognition unit. The SOPRU of Chapter IV is a direct result of this attempt.

It is my personal opinion, formed during the previously described investigation, that the correlation approach is the most promising solution to the pattern recognition problem. The correlation technique can be improved still further by combining it with the adjustable weight threshold logic approach.

It is a pleasure to acknowledge the guidance and help given to me by the following people: Captain Frank M. Brown, my faculty thesis advisor, for patiently providing guidance for my efforts; Mr. Cecil W. Gwinn of the Bionics section of the Avionics Laboratory of Wright-Patterson Air Force Base for providing reports and information; and Miss Jacqueline Collins for cheerfully and accurately translating my illegible manuscript into typewritten form.

Alling C. Foreman

Contents

Contents

List of Figures

Abstract

Most of the pattern-recognition self-organizing machines can be classified as adjustable-weight threshold-logic machines, statistical-switching machines, or correlation machines. A noisy pattern is a pattern that varies slightly from a model pattern that the machine has been taught. Computer simulations of a typical machine from each of the three classes indicate that the noisy pattern recognition capabilities are poor for statistical-switching machines, good for threshold-logic machines, best for correlation type machines. A proposed correlation self-organizing machine is simple, learns in one step, and recognizes noisy patterns accurately.

I. Introduction

One of man's greatest assets is his ability to obtain information from his environment and to make intelligent decisions concerning this information. The information is received from the environment through sense organs such as the eyes and ears. The information is then sent to the brain where it must be classified before a decision can be made to respond to the information, or stimulus, in a particular manner. This classification process is called pattern recognition.

A basic limitation of electronic computers is their ability to gain information from their environment. All information must be coded and given to the machine in a very special format. This coding process is very difficult for humans because any very slight coding error may give the computer completely false information. A possible way to overcome this language barrier is to give the machine some pattern recognition capabilities.

A computer with the ability to recognize patterns could be used to perform some of the more tedious tasks now done by humans. Thus, a typewritten page could be typed from a manuscript by a typewriter controlled by a pattern recognition machine. A machine that would be of more interest to

the Air Force and Navy would be able to spot enemy planes on a radar scope or submarines by the return pattern of a sonar signal. Such a machine could have a higher degree of reliability than humans because the machine would not get bored as a human does.

Classical logical design procedures using standard AND, OR, and NOT components have been used in some cases to develop pattern recognition devices. These devices are not flexible enough to solve general pattern recognition problems. Any slight variation in the pattern requires a completely new design.

A more flexible way to construct a pattern recognition device is to use fixed weight threshold logic. Slight variations in the pattern will not have a tendency to cause a change in the output of a threshold gate. The values of the weights may be determined by using a digital computer (Ref 7).

The weight determination procedure can be very complicated. A large change in the patterns will require a completely new set of weights. Thus, the fixed-weight threshold logic approach to the pattern recognition problem is also somewhat inflexible.

Perhaps the best method is to use self-organizing machines to solve pattern recognition problems. A self-organizing machine is sometimes called a learning machine

because it has the ability to adjust some parameter of its circuit to produce a desired output for any possible input. Thus, self-organizing machines are excellent for solving the complicated design problems of pattern recognition units.

At least three classes of self-organizing machines have been proposed in the last few years. These classes are:

1. Adjustable-weight threshold-logic machines.
2. Statistical-Switching Machines.
3. Correlation Machines.

Each of these classes of machines has certain advantages and disadvantages.

Object

The object of this thesis is to investigate and compare the pattern learning and recognition capabilities of each of the three classes of self-organizing machines. Particular emphasis will be placed on the abilities of the machines to recognize patterns that are slightly different from the model patterns that the machines have been taught to recognize.

Scope

The analysis will be restricted to combinatorial-logic machines. Thus, the recognition of sequential patterns will not be considered.

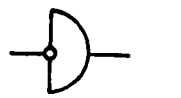
The sensor elements and the practical problems of presenting the patterns to the machine will not be discussed. It will be assumed that all sensor units perform correct measurements on the pattern, and that the outputs of the sensor units contain the information necessary to specify the value of the measurement.

Notation

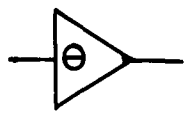
The symbols used in this report are:


AND GATE

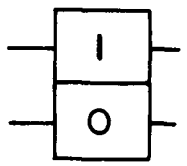

OR Gate


NOT Box


Statistical Switch



Threshold Gate with Threshold θ



Bistable Multivibrator

Σ Algebraic Summation

$+$ Logical Disjunction, OR

AND will be denoted by juxtaposition of the Boolean variables. The plus sign is used both for algebraic addition and logical disjunction, but the meaning can be found from the context. Unless otherwise specified, the weight values on the inputs of threshold gates are assumed to be one.

Organization

Each of the next three chapters will discuss one of the classes of self-organizing machines. Adjustable-weight threshold-logic machines, statistical-switching machines, and correlation machines will be described in Chapters II, III, and IV respectively. Each chapter contains a mathematical analysis of a machine, a discussion of a computer simulation of that machine, and the results of the computer simulation problems.

Chapter V consists of a comparison of the results of the investigation, conclusions, and recommendations for further study.

II. Adjustable-Weight Threshold-Logic Machines

The first self-organizing machines to be proposed for use in solving the pattern recognition problem belong to the class of learning machines that use threshold gates with adjustable weights. A single-stage device proposed by Mattson (Ref 8), and CHILD (Ref 4), an analog-input machine being built for the Air Force, will be very briefly described in this chapter. A more thorough description of a more general machine, the perceptron (Ref 9, 10, 11), will then be given. Finally, a computer simulation designed to determine experimentally some of the pattern recognition capabilities of a simple perceptron will be discussed. Before beginning the description of the machines, an explanation of some of the properties of threshold logic is in order.

Threshold Logic

Threshold logic concerns a special class of Boolean switching functions. A threshold function exists if a set of n input weights w_1, w_2,...w_i,...w_n and a threshold value t can be found which satisfy the inequalities

$$\sum_{i=1}^{n} W_i X_i \geq t \qquad (1a)$$

for all combinations of the n Boolean variables x_1, x_2,... x_i,...x_n that are logically true and

$$\sum_{n=1}^{n} w_i x_i < t \qquad (1b)$$

for all combinations of the variables that are false. For a gate to be a threshold gate two conditions must be met as follows:

1. The output of the gate must be a logical one if and only if the linear sum of the weighted inputs is greater than or equal to the threshold value.

2. The output of the gate must be a logical zero if and only if the linear sum of the weighted inputs is less than the threshold value.

An important advantage of a threshold gate over standard AND and OR type gates is the ability of the threshold gate to generate many different Boolean functions by simply varying the input weights and threshold value. Fourteen of the 16 Boolean functions of two variables can be realized by a single threshold gate. Thus, a considerable reduction in the number of components required to construct logic circuits can be made by using threshold logic. Unfortunately, the fraction of the 2^{2^n} Boolean functions of n variables that are threshold functions decreases rapidly as n

increases. However, since any Boolean function can be synthesized using AND, OR, and NOT gates, and since a threshold gate can generate these functions, any Boolean logical device can be constructed using threshold gates.

Threshold functions are sometimes called linearly separable functions. The reason for this name can be seen if the 2^n minterms of n variables are represented as vertices of an n-dimensional cube. If the vertices corresponding to true minterms can be separated from the vertices corresponding to false minterms by an n-1 dimensional hyperplane, then the function is a threshold function.

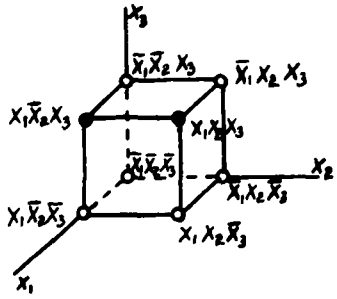
Example

Given the three variable Boolean functions:

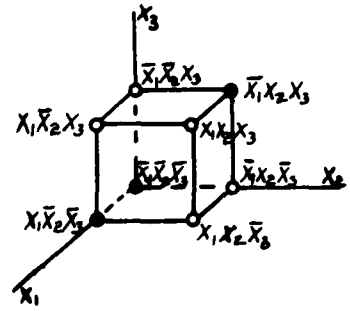
$$F_1 = x_1\bar{x}_2x_3 + x_1x_2x_3 \qquad (2)$$

$$F_2 = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 \qquad (3)$$

These equations are represented by the solid black vertices of the cubes shown below.



$F_1 = x_1\bar{x}_2x_3 + x_1x_2x_3$



$F_2 = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3$

Notice that Eq (2) is a threshold function since a 2-dimensional plane can separate the solid black vertices from the other vertices, but Eq (3) is not a threshold function since no plane can separate the solid black vertices from the rest of the vertices.

If Ineqs (1) are reduced to an equation, the resulting equation will define the n-1 dimensional hyperplane that separates the vertices representing true minterms from the vertices representing false minterms. The input weights w_i are the direction numbers of the plane and the threshold t is the squared perpendicular distance from the plane to the origin. The plane required to separate the vertices is not unique; therefore, many different combinations of weights and threshold values can be used to synthesize one Boolean logic network.

The Hamming distance between two minterms is defined as the number of edges of the n-dimensional cube that must be traveled to get from the vertex corresponding to one minterm to the vertex corresponding to the other minterm. It can be seen from the figure of the previous example that only one variable is changed in going from one vertex to the next. Thus Hamming distance is equal to the number of corresponding variables that are different in two minterms.

A characteristic of threshold functions is that minterms that are close to true minterms tend to be classified

as true minterms and minterms that are close to false minterms tend to be classified as false minterms. This is a result of the linear separability property. All minterms with vertices on one side of the plane will be classified as true and all minterms with corresponding vertices on the other side of the plane will be classified as false. Vertices far away from the plane will be with all of their close neighbors in the same classification. Vertices close to the plane will have neighbors in both classifications.

Because a threshold gate will tend to give the same response for minterms that are close to each other, threshold logic can be used to build pattern recognition devices that are relatively insensitive to slight variations in the pattern. Furthermore, the device can be constructed so that it will adjust the weights on the inputs of the threshold gates in such a manner that it will learn to recognize the patterns that are presented to it. This is the basis for the machines that will be discussed in the remaining portion of this chapter.

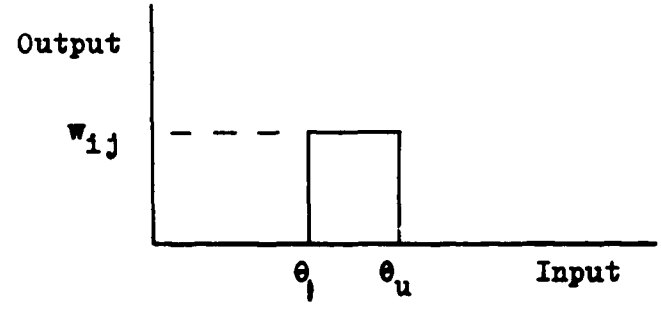
Mattson's Machine

A self-organizing binary device proposed by Mattson (Ref 8) is shown in Fig. 1. It consists of a threshold gate with continuously variable weights on the inputs and a continuously variable threshold value. A training unit samples the performance of the system and determines what

corrections to make to the weights and threshold value. Since the device can only separate minterms that are linearly separable, there is a very limited number of applications for a single stage of these devices. A multi-stage network of these learning machines can be used to realize a larger class of Boolean functions, however.

CHILD

A Cognitive Hybrid Intelligent Learning Device (CHILD), designed by Choisser (Ref 4) is presently being constructed and studied at the Rome Air Development Center in New York. The inputs to a CHILD are in the form of an analog vector with n terms. Each input is fed to a CHILD cell (Fig. 2) which has an output other than zero if and only if the value of the input is between two threshold values as shown in the transfer characteristic shown below.



The outputs of each cell are weighted and fed to the input of a threshold gate. The output of the threshold gate is a logical one if and only if the sum of the outputs of the

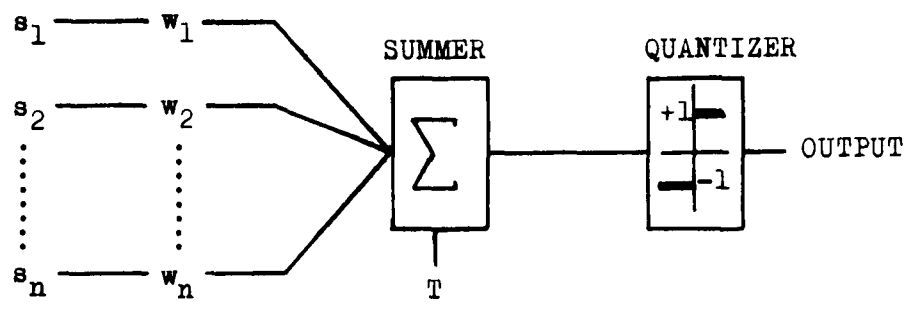


Fig. 1

An Adjustable Binary Logical Network

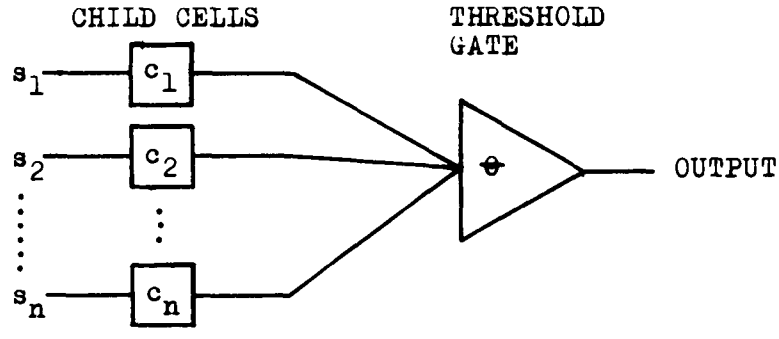


Fig. 2

A CHILD Network for One Pattern Class

CHILD cells is greater than a threshold value.

Each class of patterns to be recognized must be allotted one threshold gate and n CHILD cells. A training device will adjust the upper and lower threshold values on the inputs of the cells and the weights on the outputs of the cells until the device has learned to respond to the inputs in the desired manner.

The Perceptron

The perceptron (Ref 9, 10, 11) conceived by Rosenblatt, has been the object of research sponsored by the Navy at Cornell Aeronautical Laboratory. A simple perceptron has two stages of threshold logic. Sensor elements are connected to the first stage threshold gates by leads with weights chosen from the set $\{0, 1, -1\}$ in a random manner. The first stage outputs are weighted and fed to the second stage threshold gates. These second stage weights may be adjusted so that the second stage threshold gates will have a logical one output for some input combinations and a logical zero output for other combinations.

Mathematical Analysis. A pattern presented to a perceptron (Fig. 3) will cause the sensor (S) units to have a zero output if the sensor unit is not excited and a one output if the S unit is excited. The outputs of the sensor units can be arranged in the form of an n-element binary vector

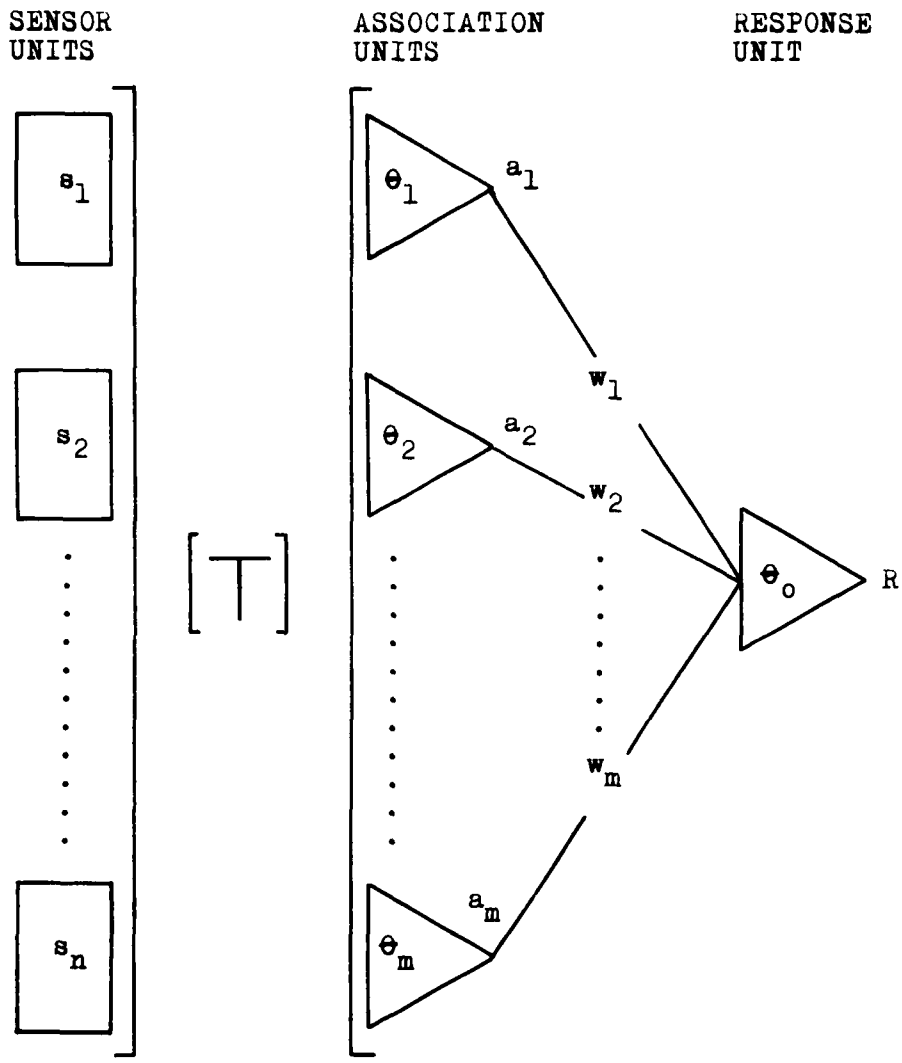


Fig. 3

A Simple Perceptron

$$\underset{\sim}{S} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \cdot \\ \cdot \\ \cdot \\ s_n \end{bmatrix} \tag{4}$$

where each s_i can have the value zero or one. This vector will often be referred to as the input vector, and will often be written in the transposed form

$$\underset{\sim}{S}^T = \begin{bmatrix} s_1 & s_2 \cdots s_j \cdots s_n \end{bmatrix} \tag{5}$$

The outputs of the sensor units are connected to association (A) units with weighted connections. These weighted connections can be arranged into a topology matrix

$$T = \begin{bmatrix} t_{11} & t_{12} \cdots t_{ij} \cdots t_{1n} \\ t_{21} & t_{22} \cdots t_{2j} \cdots t_{2n} \\ \cdot & \cdot \quad \cdot \quad \cdot \\ \cdot & \cdot \quad \cdot \quad \cdot \\ \cdot & \cdot \quad \cdot \quad \cdot \\ t_{m1} & t_{m2} \cdots t_{mj} \cdots t_{mn} \end{bmatrix} \tag{6}$$

Where each t_{ij} is the weighted connection of the jth S unit to the ith A unit, and m and n are the number of A units and S units, respectively. The value for each t_{ij} is chosen by consulting a random number table. Thus, the matrix

will be composed of a random array of zeros, ones, and minus ones.

The output of each A unit will be a logical one if and only if the sum of the weighted inputs is greater than the threshold value, and will be zero otherwise. This can be expressed as

$$A_i = \begin{cases} 1 \text{ if } \sum_{j=1}^{n} t_{ij}\, s_j \geq \theta_i \\ 0 \text{ if } \sum_{j=1}^{n} t_{ij}\, s_j < \theta_i \end{cases} \tag{7}$$

where A_i is the output and θ_i is the threshold value of the ith A unit.

The outputs of the A units are multiplied by the adjustable weights and summed at the input of the response (R) unit. The output of the response unit, which is the output of the perceptron, will be a logical one if the sum of the weighted outputs of the A units is greater than or equal to the threshold value, and will be zero otherwise. Thus,

$$R = \begin{cases} 1 \text{ if } \sum_{i=1}^{m} w_i a_i \geq \theta_o \\ 0 \text{ if } \sum_{i=1}^{m} w_i a_i < \theta_o \end{cases} \tag{8}$$

where w_i is the weight of the connection of the ith A unit to the R unit, and θ_o is the threshold value of the R unit threshold gate.

Number of Units Required. The number of sensor units required for a pattern recognition problem depends upon the complexity of the patterns. More details of a pattern can be used to classify the pattern if a higher number of sensors is used. In the Mark I perceptron built at Cornell, 400 sensor units were used in a 20 x 20 array (Ref 10). Plans for the Mark II perceptron indicated that approximately 4000 sensor units would be used.

The number of association units required for the perceptron depends upon the number of sensor units used. Ideally, 2^n A units should be used to insure that the A unit output vectors corresponding to the input vectors of the patterns to be learned are not logically close in terms of Hamming distance. Note that no adjustment of the input weights of the R unit will allow the perceptron to distinguish between two input vectors with the same A unit responses. The number 2^n gets extremely large as n increases, however, so a more realistic number of A units must be used. The number of A units used in the Mark I perceptron was only 512, but the number planned for the Mark II approached 100,000.

The number of R units required depends upon the

number of different classifications to be learned. One R unit could be used for each class, but this would be inefficient because the output can be encoded with a much smaller number. In fact, the number of R units only needs to be greater than or equal to $\log_2 k$ where k is the number of classifications to be made, for a completely different binary code for each class. Thus, six R units are required to encode the alphabet and ten numbers, since six is equal to $\log_2 64$ which is greater than $\log_2 62$.

Organization Procedure. Several training methods have been investigated for the simple perceptron at Cornell. The most successful of these is the α-error correction procedure. In fact it has been proven that this procedure will yield a solution to the pattern classification problem in finite time, provided each pattern is presented again in finite time in any sequence, and provided a solution exists (Ref 11:18). For a solution to exist, the responses of the A units to the various input vectors must be linearly separable.

The α-error correction system changes the weights on the R-unit input connections only if the perceptron response is not correct, and only if the A unit associated with the weighted connection had a logical one output. In general, the value of the change can be variable and need not be an integer. For the computer simulations to be described, the value of the weight change will be a

constant equal to one. Thus, the correction to the ith weight can be formulated as

$$(w_i)_{new} = (w_i)_{old} + a_i(R_{desired} - R_{actual}) \qquad (9)$$

where w_i is the weight of the connection of the ith A unit to the R unit, a_i is the output of the ith A unit, $R_{desired}$ is the desired response of the perceptron to the input vector, and R_{actual} is the actual response of the perceptron.

It has been shown that the order of presentation of the input vectors to the perceptron does not affect the ability of the perceptron to learn to recognize the patterns (Ref 11:18). It will affect the learning time, however. In the computer simulation described later in the chapter, the patterns were presented so that the desired responses were an alternating sequence of zeros and ones. After all inputs were presented, the sequence was repeated.

Noisy Patterns. For a pattern recognition device to be of any value, it must be capable of recognizing a pattern even though there is a slight variation in the pattern. Slight variations of the input vector from a model vector which the perceptron has learned to recognize will be termed noise. Thus, if one digit of the input vector is different from the corresponding digit of the model vector, the pattern will be said to have one digit noisy.

Computer Simulation of a Simple Perceptron

It was decided that a good way to determine the noisy pattern recognition capabilities of a simple perceptron would be to work through several example problems to see how well the perceptron classified the noisy patterns. Since hand calculations of this sort would be extremely tedious, several computer programs were written by the author to simulate the perceptron. A typical program for an IBM 1620 digital computer is discussed in Appendix A.

The perceptron simulated for the first two problems had 25 sensor units, 50 association units, and one response unit. The number of sensor units was reduced to nine for the third problem. All threshold values and the initial values of the weights were zero. Instead of consulting a random number table to obtain a random topology ($[T]$) matrix, a short machine language program was written to generate a random array of zeros, ones, and minus ones, and to punch the array onto IBM cards. Another machine language program was used to translate model input vectors into noisy input vectors and to punch the noisy vectors onto cards.

The remaining portion of this chapter will be used to discuss three pattern recognition problems and the results of the problems.

Problem 1. The first problem to be presented to

the perceptron simulation program was to learn to place the letters A, B, C, D, and E into one classification, and the numbers 1, 2, 3, 4, and 5 into another classification. The desired response for the letters was arbitrarily chosen to be a logical one, and the desired response for the numbers was chosen to be a logical zero. The model letters and numbers are shown on the 5 x 5 grids of Fig. 4. The dark squares correspond to sensor units that have a logical one output, and the light squares correspond to sensor units that have logical zero outputs. The input vectors were coded from the patterns by scanning each row from left to right, starting with the top row, and placing a zero in the input vector for each light square and a one for each dark square. The numbers in the grid below are the positions of the elements of the grid in the input vectors.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

The model input vectors for the letters of Fig. 4 are shown below in the order that they were presented to the perceptron.

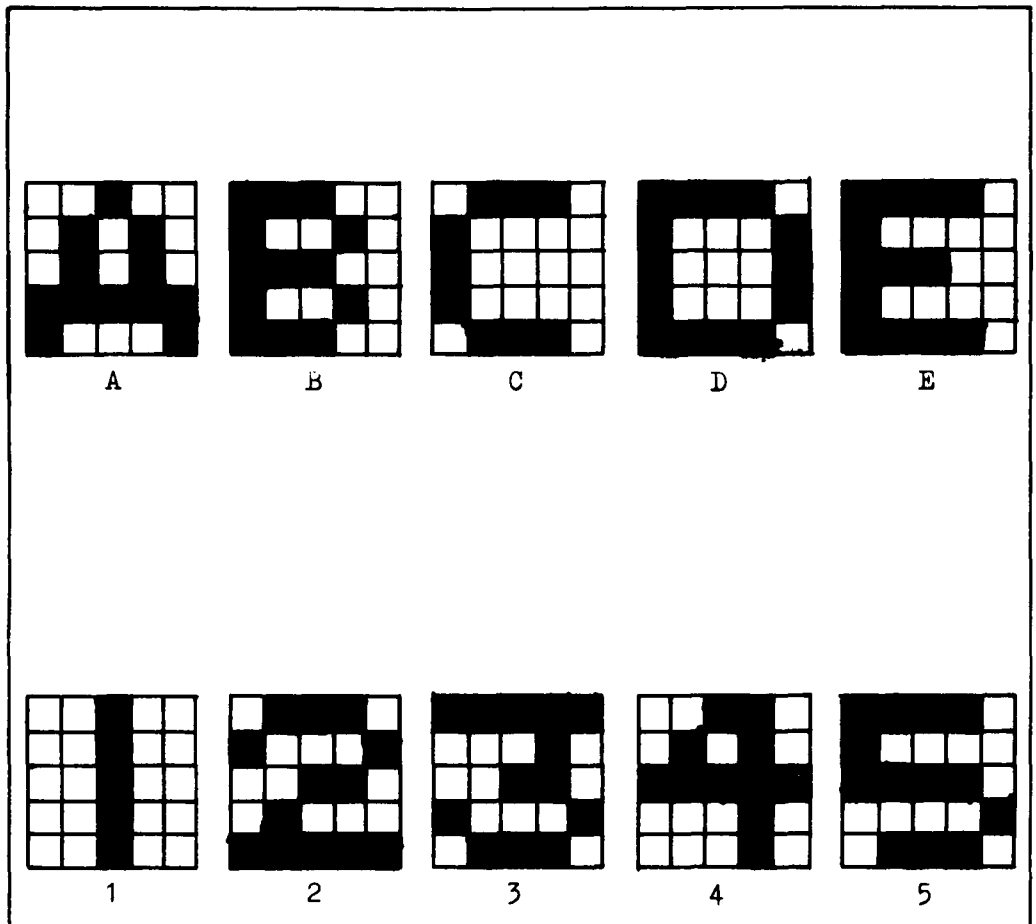


Fig. 4

Problem 1 Input Patterns

$$
\begin{aligned}
\underset{\sim}{S}_A^T &= 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1 \\
\underset{\sim}{S}_1^T &= 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\
\underset{\sim}{S}_B^T &= 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \\
\underset{\sim}{S}_2^T &= 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \\
\underset{\sim}{S}_C^T &= 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_3^T &= 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_D^T &= 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \\
\underset{\sim}{S}_4^T &= 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0 \\
\underset{\sim}{S}_E^T &= 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_5^T &= 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0
\end{aligned}
\qquad (10)
$$

After the perceptron had learned to correctly classify the model input vectors, 500 samples of noisy input vectors were presented to the computer to find out if they would be classified correctly. No weight adjustments were made during these tests. Each of the digits of each of the model input vectors was changed only once to generate 250 input vectors with one digit noisy. Two digits were perturbed in each of the last 250 noisy input vectors.

Results of Problem 1. The perceptron learned to properly classify each of the 10 model vectors after each vector had been presented to the machine only twice.

The perceptron made 22 mistakes in classifying the

Character	Number of Mistakes	Recognition Level	Critical Digits
A	5	80%	9, 13, 16, 19, 21, 23
B	1	96%	11
C	3	88%	2, 7, 16
D	0	100%	
E	2	92%	20, 25
1	0	100%	
2	0	100%	
3	2	92%	13, 24
4	7	72%	1, 4, 5, 10, 14, 20, 24
5	1	96%	21

Table I

Results of One-Noisy Digit Tests of Problem 1

Character	Number of Mistakes	Recognition Level	2 Critical Digits
A	7	72%	1, 21; 13, 12; 14, 18; 16, 13; 19, 24; 23, 15; 24, 16
B	3	88%	2, 20; 9, 16; 24, 15
C	1	96%	2, 18
D	0	100%	
E	6	76%	2, 17; 4, 14; 5, 2; 16, 9; 17, 10; 18, 11
1	0	100%	
2	0	100%	
3	5	80%	5, 4; 7, 10; 8, 21; 13, 24; 19, 1
4	11	56%	2, 10; 5, 17; 6, 18; 7, 21; 10, 24; 14, 20; 15, 18; 18, 2; 20 5; 21, 6; 24, 25
5	4	84%	1, 13; 13, 25; 19, 5; 24, 17

Table II

Results of Two-Noisy Digit Tests of Problem 1

250 vectors with one noisy digit, so the correct response was given for 91.3% of the samples. Table 1 shows the number of mistakes made in classifying each letter and number, and which digits were noisy when the mistakes were made.

Thirty-seven of the 250 input vectors with two digits noisy were not classified correctly. Thus, the overall recognition level was 85% for this case. A listing of the mistakes is shown in Table 2.

Problem 2. For the second problem the number of pattern classes was increased to 25. The perceptron was required to learn to classify the Arabic letters in Fig. 5 with a logical one response, and the numbers and Greek letters with a logical zero response. These characters were chosen from a group of 50 tested because they seemed to be the most difficult for the machine to classify. The input vectors are shown below in the order that they were presented to the perceptron.

$\underset{\sim}{S}_A^T$ = 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

$\underset{\sim}{S}_1^T$ = 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0

$\underset{\sim}{S}_B^T$ = 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0

$\underset{\sim}{S}_3^T$ = 1 1 1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0

$\underset{\sim}{S}_D^T$ = 1 1 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0

$\underset{\sim}{S}_4^T$ = 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0 0 1 0

$$
\begin{aligned}
\underset{\sim}{S}_{E} &= 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \\
\underset{\sim}{S}_{5} &= 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_{G} &= 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_{7} &= 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\
\underset{\sim}{S}_{H} &= 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \\
\underset{\sim}{S}_{O} &= 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_{I} &= 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \\
\underset{\sim}{S}_{L} &= 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \\
\underset{\sim}{S}_{Q} &= 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1 \\
\underset{\sim}{S}_{\wedge} &= 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1
\end{aligned}
\qquad (11)
$$

$$
\begin{aligned}
\underset{\sim}{S}_{S} &= 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \\
\underset{\sim}{S}_{W} &= 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \\
\underset{\sim}{S}_{Y} &= 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\
\underset{\sim}{S}_{Z} &= 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0 \\
\underset{\sim}{S}_{K} &= 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \\
\underset{\sim}{S}_{\$} &= 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\
\underset{\sim}{S}_{N} &= 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \\
\underset{\sim}{S}_{\#} &= 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \\
\underset{\sim}{S}_{\dagger} &= 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0
\end{aligned}
$$

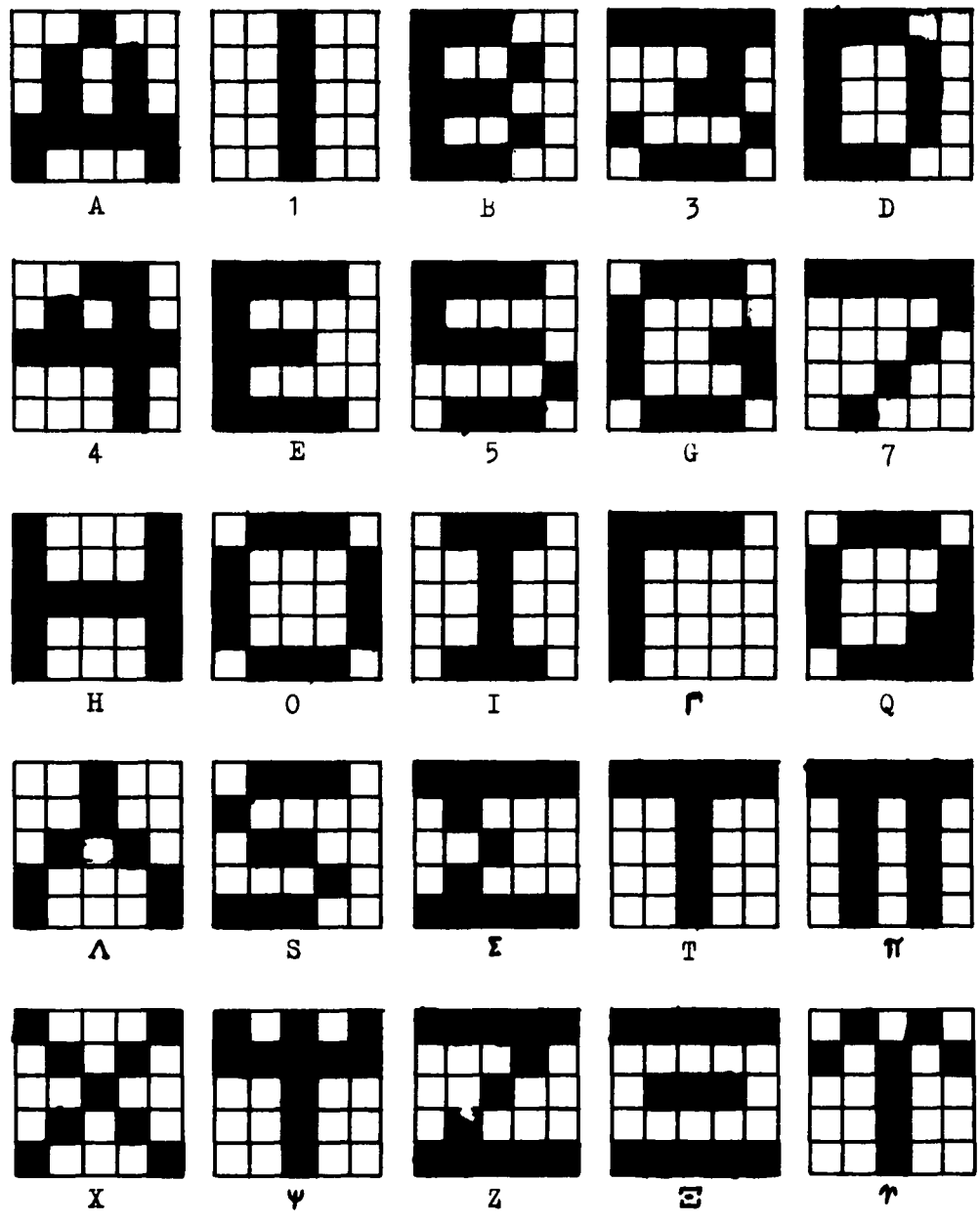


Fig. 5

Problem 2 Input Patterns

As in the previous problem, the weight adjustments were discontinued after the perceptron had learned to recognize the model patterns, and noisy input vectors were presented. All of the 625 possible vectors with one noisy digit and 600 of the 7500 possible input vectors with two noisy digits were shown to the machine.

Results of Problem 2. Each input vector had to be presented to the perceptron 32 times before the machine learned to recognize the patterns.

Of the 625 input vectors with one digit noisy, 92 were given the wrong classification. Thus the overall recognition level was 85%. Table 3 is a listing of all of the errors made.

Only 75% of the input vectors with two noisy digits were recognized. There were 149 mistakes made in classifying the samples. The mistakes are tabulated in Table 4.

Problem 3. For the third problem, the number of sensor (S) units was reduced to nine. This reduction was made so that the results of the problem could be compared to the results obtained from a simulation of the Self-Organizing Binary Logical Network described in the next chapter. Because 50 association units were used for this problem as well as the first two problems, a qualitative analysis of the effects of increasing the ratio of association units to sensor units can also be made.

Character	Number of Mistakes	Recognition Level	Critical Digits
A	5	80%	4, 8, 11, 13, 15
1	5	80%	1, 16, 19, 20, 22
B	0	100%	
3	7	72%	6, 7, 10, 12, 18, 21, 25
D	2	92%	4, 24
4	0	100%	
E	4	84%	9, 15, 20, 25
5	0	100%	
G	7	72%	5, 7, 10, 13, 17, 18 21
7	1	96%	12
H	2	92%	4, 9
0	2	92%	17, 19
I	1	96%	14
	8	68%	7, 8, 9, 17, 18, 19, 22, 25
Q	2	92%	1, 13
	9	64%	4, 5, 6, 7, 11, 18, 19, 22, 23
S	1	96%	10
	10	60%	6, 8, 9, 11, 14, 15, 16, 18, 19, 20
T	4	84%	11, 12, 14, 15
	3	88%	15, 18, 21
X	5	80%	3, 4, 10, 23, 24
	1	96%	19
Z	6	76%	6, 11, 14, 15, 19, 20
	0	100%	
	7	72%	1, 5, 7, 16, 21, 22 25

Table III

Results of One-Digit Noise Tests of Problem 2

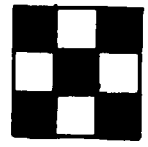
Vector Number	Character	Number of Mistakes	Recognition Level	2 Critical Digits*
1	A	1	96%	11
2	1	20	17%	*All except 10, 12, 14, 21
3	B	1	96%	14
4	3	7	71%	7, 8, 10, 12, 18, 21, 25
5	D	1	96%	24
6	4	0	100%	
7	E	0	100%	
8	5	7	71%	7, 16, 17, 18, 19, 21, 25
9	G	6	75%	1, 5, 10, 12, 17, 19
10	7	5	79%	6, 8, 12, 16, 25
11	H	2	92%	4, 9
12	O	3	88%	5, 18, 25
13	I	0	100%	
14		3	88%	8, 19, 22
15	Q	4	83%	1, 9, 13, 17
16		11	54%	2, 4, 5, 6, 7, 11, 16, 18, 19, 22, 23
17	S	0	100%	
18		23	4%	*All except 11
19	T	2	92%	11, 14
20		22	8%	*All except 11, 21
21	X	4	83%	3, 4, 10, 23
22		5	79%	7, 17, 19, 24, 25
23		6	75%	6, 10, 11, 14, 15, 19
24		5	79%	6, 7, 8, 19, 20
25		15	38%	*All except 1, 9, 11, 12, 14, 16, 19, 20, 21, 22

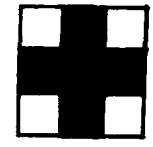
Table IV

Results of Two-Digit Noise Tests of Problem 2

*The digit corresponding to the vector number was always noisy in these tests, so these digits were not included in this column. Thus, vectors particularly sensitive to this digit were almost always incorrectly classified.

Because there are very few letters that can be formed on a 3 x 3 grid, the characters x and + were used. These symbols have only one element of their input vectors in common. The patterns and their input vectors are shown below.





$\underset{\sim}{S}_x^T = 101010101$ $\underset{\sim}{S}_+^T = 010111010$

After the perceptron had learned to recognize these patterns, the weight adjustments were stopped, and the perceptron was asked to classify all of the 18 possible patterns with one digit perturbed.

Results of Problem 3. Six different random topology ([T]) matrices were used to simulate six different perceptrons. Each of the perceptrons learned to correctly identify the x and the + after seeing each vector only once. Four of the perceptrons classified all 18 noisy patterns correctly. Each of the other two perceptrons incorrectly classified one of the 18 vectors with one digit changed. Thus, four of the perceptrons had a 100% recognition level, and two had a 94% recognition level.

Discussion of Problem Results. The results of the

simulation experiments support the following qualitative theories:

1. An increase in the ratio of association units to sensor units tends to speed up the learning process and to increase the noisy pattern recognition capabilities of a simple perceptron.

2. An increase in the number of classes of patterns to be recognized will tend to slow down the learning process and decrease the noisy pattern recognition capabilities of a simple perceptron.

3. The pattern recognition capabilities of a simple perceptron are functions of the topology matrix. Thus, some perceptrons will perform better than others on a given set of patterns.

It should be recalled that at least $\log_2 k$ response units are required to completely specify the responses to k pattern classes. Thus, for problem 2, five response units would be required to function properly in order to differentiate between each of the 25 pattern classes.

III. Statistical-Switching Machines

Another approach to the design of self-organizing machines is to use standard logical devices such as AND, OR, and NOT gates connected together by statistical switches. Statistical switches have a variable probability of being closed at any given time. Research on this class of machines has been conducted for the Air Force by Melpar, Incorporated. Several devices have been built for test purposes.

Several methods for connecting the gates and switches have been investigated. One approach is to use a basic building block called an Artron (Ref 3), a two-input one-output network that is capable of learning any of the 16 switching functions of two variables. The Artrons may be connected together in a completely random network, or they may be connected in a systematic manner. In either case the networks cannot be analyzed except on a statistical basis.

The Self-Organizing Binary Logical Network (SOBLN) can be studied much more easily. A SOBLN can learn any of the 2^{2^n} Boolean functions of n variables, which is not necessarily true of Artron networks. Furthermore, the number of statistical switches required to construct a

completely general SOBLN is a minimum (Ref 3:97).

Since most studies of a Self-Organizing Binary Logical Network can be applied to Artron networks (Ref 3:97), the SOBLN will be the only machine discussed in detail in this chapter. First, the operation of these machines will be explained, and then a computer simulation of a SOBLN will be discussed.

A Self-Organizing Binary Logical Network

Number of Components Required. A completely general Self-Organizing Binary Logical Network, one that can learn any of the 2^{2^n} Boolean functions, must have as many AND gates as there are minterms of the n input variables. Thus, a generalized SOBLN must have 2^n AND gates.

As in the perceptron, there must be at least $\log_2 k$ outputs to completely specify k pattern classes. For each output there must be one OR gate and 2^n statistical switches to connect the AND gates to the OR gates. Thus, a SOBLN must have at least $\log_2 k$ OR gates and $2^n \log_2 k$ statistical switches.

Every SOBLN must have as many NOT gates as there are sensor (input) units. A SOBLN with n sensor units, n NOT gates, 2^n AND gates, 2^n statistical switches, and one OR gate is shown in Fig. 6.

Analysis. For each set of inputs corresponding to a pattern, the outputs of the sensor units of the SOBLN may

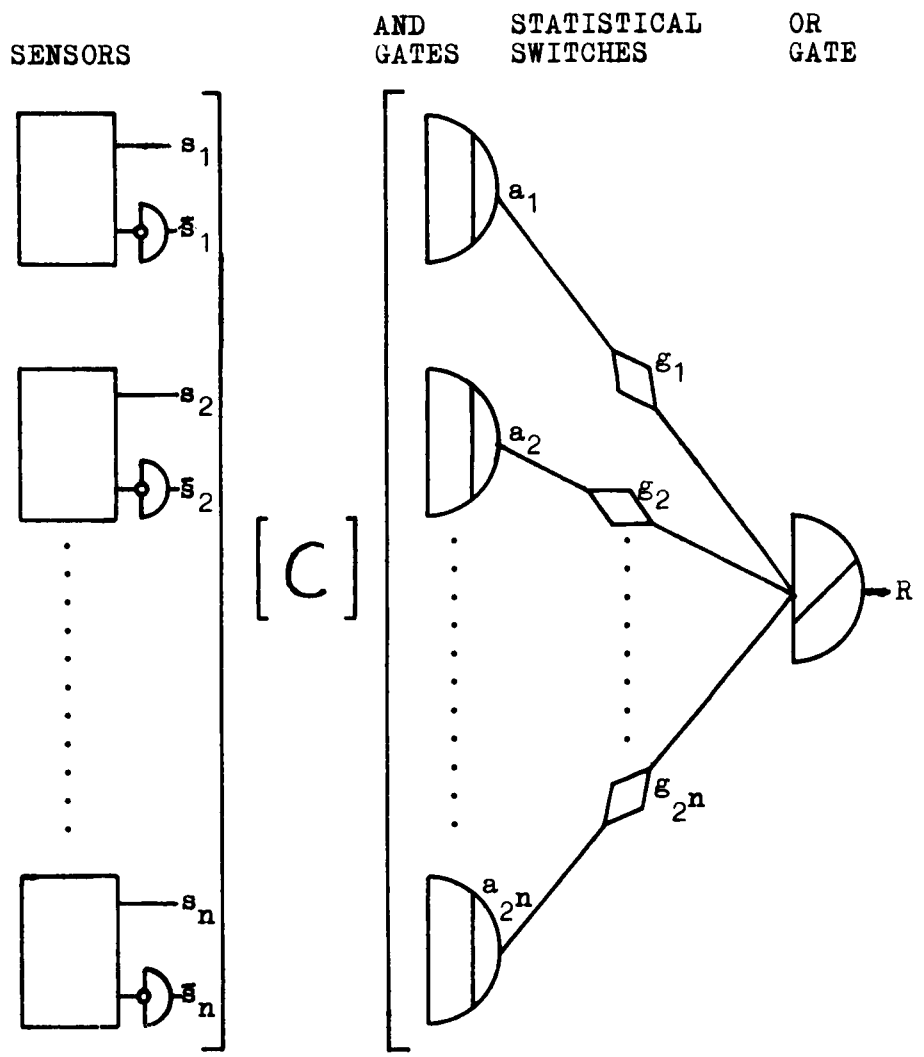


Fig. 6

A Self-Organizing Binary Logical Network

be arranged in the form of a binary vector

$$\underset{\sim}{S} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix} \tag{12}$$

where s_i is the output of the ith sensor unit. As in the analysis of the perceptron, this vector will be called the input vector and will often be written in its transposed form

$$\underset{\sim}{S}^T = \begin{bmatrix} s_1 & s_2 & s_3 \cdots s_j \cdots s_n \end{bmatrix} \tag{13}$$

The output of each sensor unit is connected to an inverter so that both s_i and its logical complement, $\bar{s}_i$, are available.

The sensor units and inverters are connected to the AND gates by a network that can be described by a connection matrix

$$C = \begin{bmatrix} c_{11} & c_{12} \cdots c_{1j} \cdots c_{1n} \\ c_{21} & c_{22} \cdots c_{2j} \cdots c_{2n} \\ \vdots & \vdots \quad \vdots \quad \vdots \\ c_{i1} & c_{i2} \cdots c_{ij} \cdots c_{in} \\ \vdots & \vdots \quad \vdots \quad \vdots \\ c_{m1} & c_{m2} \cdots c_{mj} \cdots c_{mn} \end{bmatrix} \tag{14}$$

where each c_{ij} is the connection of the jth sensor unit or inverter to the ith AND gate. The value of c_{ij} may be zero or one. If c_{ij} is one, the connection is made directly to the sensor unit. If c_{ij} is zero, the connection is made to the inverter. The connection matrix may be arranged in the form of an n-variable truth table with each of the 2^n rows corresponding to one of the 2^n minterms or canonical products of the n inputs.

Because each AND gate is connected in the manner described by the connection matrix of Eq (14), only one AND gate will have a logical one response for any given input vector. All other AND gate responses will be a logical zero. Thus, the response of the ith AND gate can be written as

$$a_i = \begin{cases} 1 \text{ if } \underset{\sim}{S}^T = \underset{\sim}{C}_i \\ 0 \text{ if } \underset{\sim}{S}^T = \underset{\sim}{C}_i \end{cases} \qquad (15)$$

where a_i is the response of the ith AND gate, $\underset{\sim}{S}^T$ is the input vector and $\underset{\sim}{C}_i$ is the ith row vector of the connection matrix.

Each AND gate is connected to the output OR gate by a statistical switch. The only statistical switch that can possible have a logical one output is the switch that is connected to the AND gate that had a logical one output. That switch will have a logical one output only if it happened to be closed at the time that the AND gate response

occurred. If a statistical switch has a logical one output, the OR gate will have a logical one output. If the state of a switch at any given time is denoted as

$$g_i = \begin{cases} 0 \text{ if the ith statistical switch is open} \\ 1 \text{ if the ith statistical switch is closed} \end{cases} \quad (16)$$

where g_i is the state of the ith statistical switch, the response of the SOBLN to an input vector is

$$R = \begin{cases} 1 \text{ if } a_i=1 \text{ and } g_i=1 \\ 0 \text{ if } a_i=0 \text{ and } g_i=0 \end{cases} \quad (17)$$

Organization Procedure. The method for organizing a SOBLN involves adjusting the closure probabilities of the statistical switches after each input vector is presented. Only the probability of the switch associated with the AND gate which had a logical one output is adjusted (Ref 3:16). The probabilities of the statistical switches are initially set to be 0.5. If the desired response is a logical one, the probability may be increased to a maximum of 1.0. If the desired response for the input vector is zero, the probability may be decreased to a minimum of zero.

The rules for adjusting the switch probability are (Ref 3:99):

1. If the statistical switch was closed, a reward increases the probability and a punishment decreases the

closure probability.

2. If the statistical switch was open, a reward decreases the probability and a punishment increases the closure probability.

These rules can be simplified using Boolean algebra. Let

A_i = response of the ith AND gate to last input vector

U = increase probability of statistical switch

D = decrease probability of statistical switch

G = switch was closed

$\overline{G}$ = switch was open

P = punish switch

$\overline{P}$ = reward switch

Then

$$U = A_i(G\overline{P} + \overline{G}P) \tag{18}$$

and

$$D = A_i(GP + \overline{G}\overline{P}) \tag{19}$$

But P and $\overline{P}$, the punish and reward variables, can be written in terms of the desired response. Let

R_d = desired response is 1

$\overline{R}_d$ = desired response is 0

Then

$$P = (A_iG)\overline{R}_d + (\overline{A_iG})R_d \tag{20}$$

and

$$\overline{P} = (A_i G)R_d + (\overline{A_i G})\overline{R}_d \tag{21}$$

Substituting Eqs (20) and (21) for P and $\overline{P}$, Eqs (18) and (19) become

$$U = A_i\left\{G\left[(A_i G)R_d + (\overline{A_i G})R_d\right] + \overline{G}\left[(A_i G)\overline{R}_d + (\overline{A_i G})R_d\right]\right\} \tag{22}$$

$$D = A_i\left\{G\left[(A_i G)\overline{R}_d + (\overline{A_i G})R_d\right] + \overline{G}\left[(A_i G)R_d + (\overline{A_i G})\overline{R}_d\right]\right\} \tag{23}$$

Simplifying, Eqs (22) and (23) can be written as

$$U = A_i R_d \tag{24}$$

$$D = A_i \overline{R}_d \tag{25}$$

Thus, the rules can be reduced to the following statements:

1. If the desired response is a logical one, and the ith AND gate response is one, increase the closure probability of the ith statistical switch.

2. If the desired response is zero, and the ith AND gate response is one, decrease the closure probability of the ith statistical switch.

For strictly combinatorial pattern recognition problems, the probabilities can be changed immediately to zero or one, depending upon the desired response. If the problem is of a sequential nature, the probability should be changed in small increments.

Noisy Pattern Recognition. It can be seen from the following statements that the probability of recognizing a noisy pattern correctly is 0.5. Only the probabilities of the statistical switches associated with the model input vectors are adjusted during the training process. If a noisy pattern is presented, the AND gate which has a logical one response is connected to the OR gate through an untrained statistical switch. Since the untrained statistical switch has a closure probability of 0.5, the recognition level will approach 50% when many noisy input vectors are shown to the machine or when one noisy pattern is repeatedly given to the SOBLN.

A partial SOBLN could be constructed by using less than 2^n AND gates and statistical switches. If an input vector that was not equal to one of the row vectors of the connection matrix were presented to the partial SOBLN then none of the AND gates would have a logical one response. Because none of the AND gates would have a logical one response, the output of the SOBLN would always be zero for any such input vector. If it is assumed that there are equally as many desired responses equal to one as there are desired zero responses for a large set of input vectors, then the probability of correctly identifying an input vector for which there is no corresponding AND gate approaches 0.5. Indeed, it appears that if the responses

for all of the possible input vectors for any logical network are not specified in the design, the probability of correctly responding to an unspecified vector will approach 0.5.

Sequential Pattern Recognition. A job that the SOBLN is better suited for is the recognition of patterns in sequences of events. Thus, the SOBLN may learn to anticipate the next number of a series of numbers, the correct turn for a mechanical rat to make when running a maze, or to play a competitive game with a biased opponent. This theory is supported by the study of the type of games described in Ref 3.

Computer Simulation of a SOBLN

A computer program to simulate a Self-Organizing Binary Logical Network was written by the author for an IBM 1620 digital computer. Because over 33 million AND gates and statistical switches would be required for a 25-input SOBLN, the first two problems given to the perceptron could not be simulated. The number of sensor units was reduced to nine so that only 512 AND gates and statistical switches were necessary. As in the perceptron experiments, only one output unit was used.

The statistical switch probabilities could be varied from 0.50 to either 0.00 or 0.99 in six steps by changing the value of a counter. The counter values and their

corresponding probability values are shown in the table below.

Counter Value	Probability That Statistical Switch is Closed
0	0.00
1	0.01
2	0.03
3	0.06
4	0.12
5	0.25
6	0.50
7	0.75
8	0.87
9	0.93
10	0.96
11	0.98
12	0.99

Noisy Pattern Recognition Problem. Problem 3 of Chapter II was also presented to the SOBLN simulation program. This problem required the SOBLN to learn to classify the characters x and + with a logical zero and a logical one respectively. The input vectors corresponding to these characters are repeated below.

$$\underset{\sim}{S}_x^T = 101010101$$
$$\underset{\sim}{S}_+^T = 010111010 \qquad (26)$$

After the SOBLN had learned to recognize these patterns, all of the 18 possible patterns with one noisy digit were

presented to the machine. These same 18 patterns were shown to the SOBLN 23 times for a total of 414 trials.

Results. The SOBLN learned to correctly identify the model input vectors after seeing each vector six times. The machine failed to identify correctly 202 of the 414 noisy input vectors. Thus, the recognition level was approximately 51%. Both of these results support the theoretical analysis of the Self-Organizing Binary Logical Network. It is interesting to note that Buroker (Ref 2) obtained similar results with the same data on an existing statistical-switching machine.

IV. Correlation Machines

Another approach to the pattern recognition problem is to use correlation techniques. These techniques are discussed by Highleyman (Ref 6) and Fischler (Ref 5).

The correlation of two vectors is a measure of the agreement of the two vectors. The degree of correlation between two n-component binary vectors is defined here to be the number of corresponding components of the two vectors that agree. Thus, the degree of correlation of two equal vectors is n. The degree of correlation between two unequal vectors is n minus the Hamming distance between the two vectors.

The assumption has been made in this report that noisy patterns should be classified in the same class as the model pattern that is the closest in terms of Hamming distance to the noisy pattern. A correlation pattern recognition device measures the degree of correlation of the input vector to the model vectors, determines the maximum degree of correlation, and then classifies the input vector as a member of the same class as the model vector with the maximum degree of correlation with the input vector.

Very few correlation type self-organizing machines have been presented in the literature. The Steele Neuron

(Ref 1), an analog device, has been proposed for use in pattern recognition systems. The Self-Organizing Pattern Recognition Unit (SOPRU), which is the only device that will be discussed in this chapter, is proposed by the author.

A Self-Organizing Pattern Recognition Unit

Number of Components Required. The Self-Organizing Pattern Recognition Unit shown in Fig. 7 must have n sensor units, n NOT boxes, m minterm recognition units, k OR gates, and one control unit, where n is the number of inputs, m is the number of model input vectors, and k is the number of pattern classes. Each minterm recognition unit (Fig. 8a) contains n single-pole double-throw locking relays, one 2n-pole single-throw switch, one summer, and one threshold gate. The electronic realization of the minterm recognition unit (Fig. 9) consists of n bistable multivibrators, 3n AND gates, one summing device, and one threshold gate. The control unit (Fig. 8b) is composed of a MAX gate, a polarity changer, and two threshold gates.

Organization Procedure. Each minterm recognition unit is individually trained to recognize one model input vector. This is accomplished by closing the learning control switch in the minterm recognition unit to be trained and presenting the model pattern to the sensor units. Each sensor unit will have a logical one output if it is excited and a

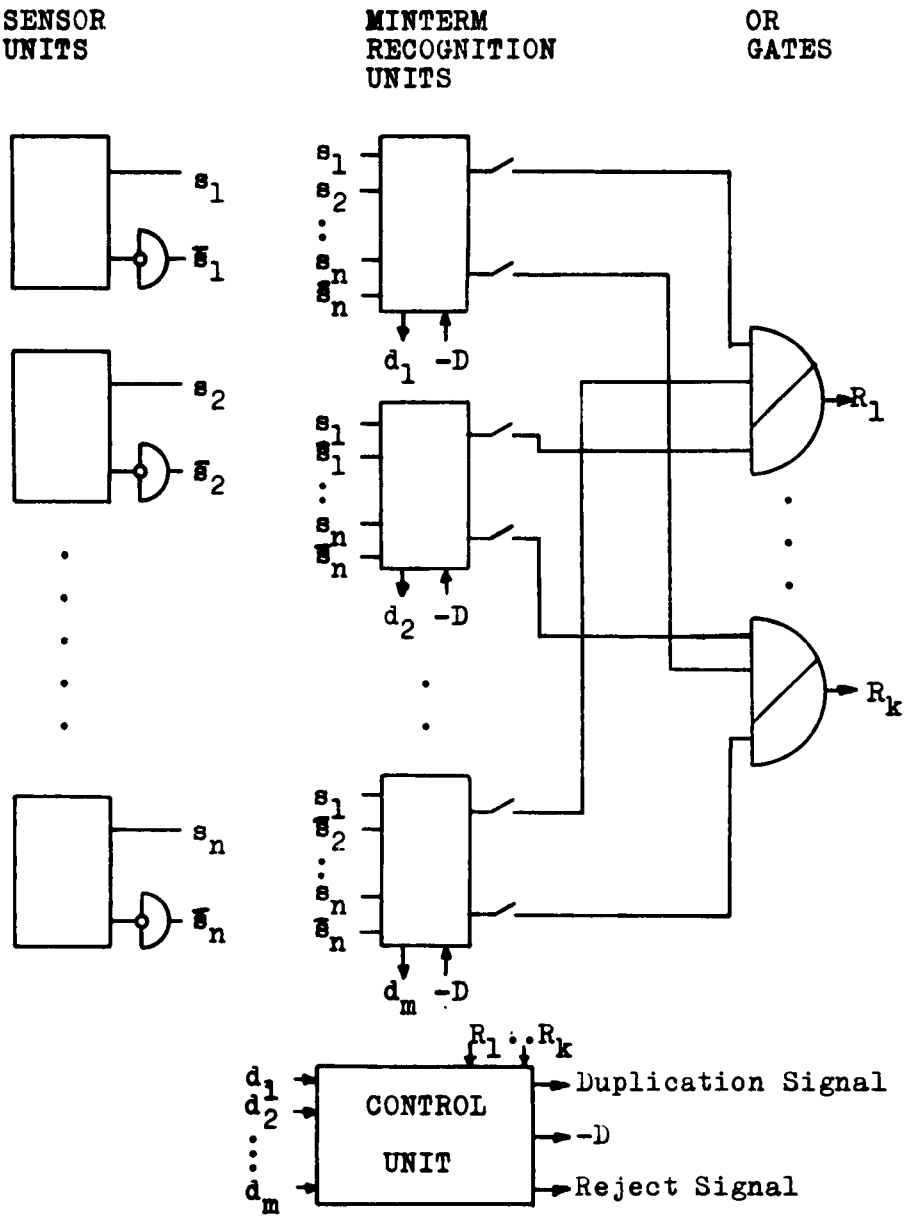


Fig. 7

A Self-Organizing Pattern Recognition Unit

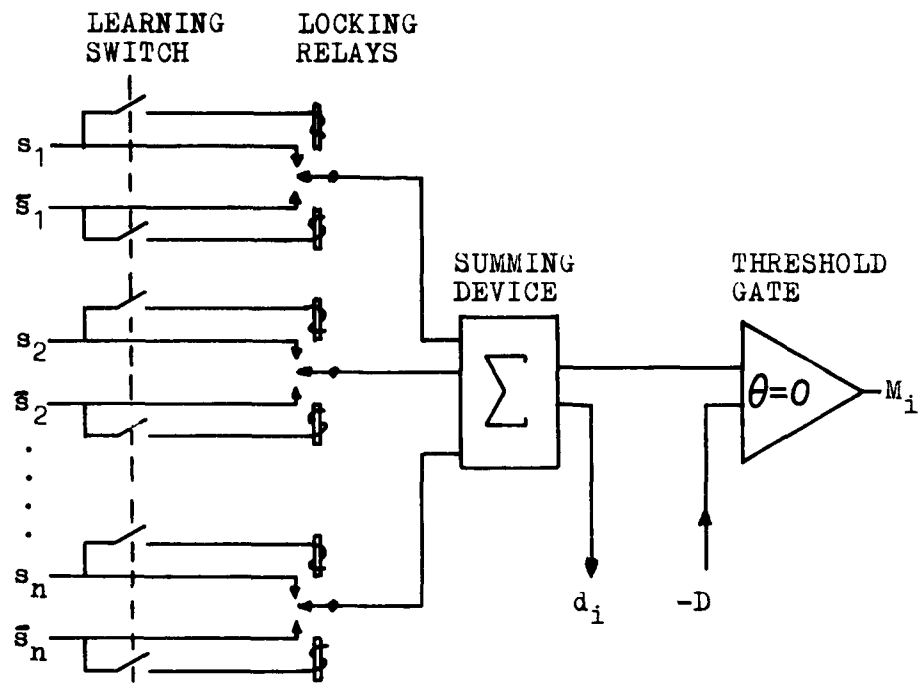


Fig. 8 a

A Minterm Recognition Unit

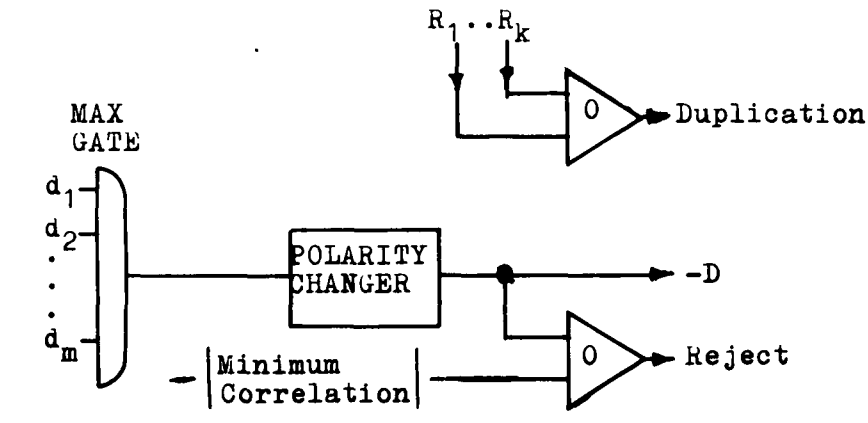


Fig. 8b

Control Unit

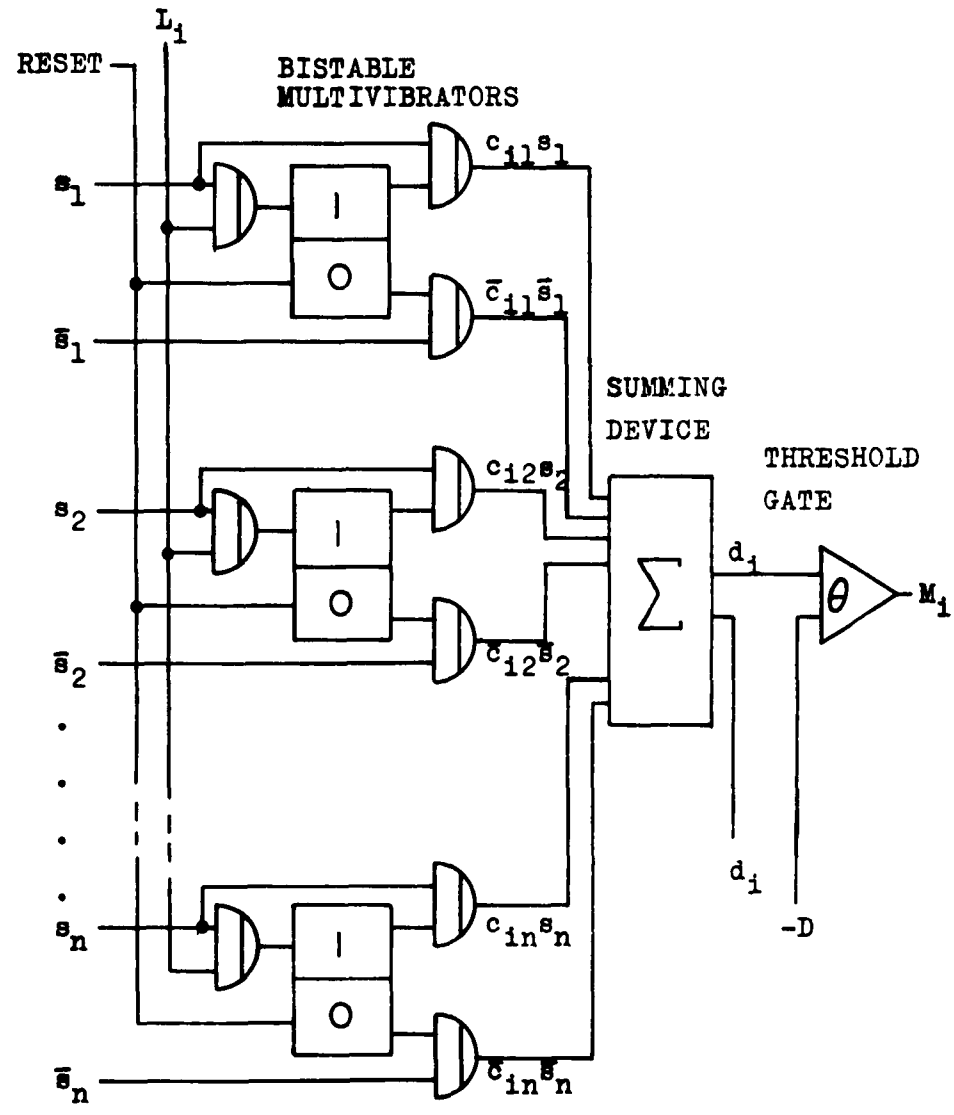


Fig. 9

A Minterm Recognition Unit

with Electronic Components

zero output otherwise. If the output is zero, the NOT box will have a logical one output. Therefore, either the sensor unit or the inverter will always have a logical one output, and the corresponding coil of the locking relay will be excited. The relay will switch to the contact with the excited coil and lock in that position.

Because all relays will be set simultaneously, the learning process for each model vector is completed in one step. The learning control switch should then be opened. The next model vector can then be taught to the next minterm recognition unit. After all minterm recognition units have been trained, the organization procedure is complete. It should be noted that all of the information necessary to specify a model vector is contained in the position of the locked relays in a minterm recognition unit.

Pattern Recognition Process. When a pattern is presented to the SOPRU after the organization process has been completed, the relays will remain in the same position because all of the learning control switches are open. Therefore, in each minterm recognition unit, the only summing device input leads that will have a signal will be those which are connected through the relay to a sensor unit or inverter that has a logical one output. Because the positions of the relays were determined from the model input vectors during the training process, the output of

the summing device will be a direct measurement of the degree of correlation between the input vector and the minterm recognition unit model vector.

As in the previous chapter, the outputs of the sensor units can be arranged in the form of a column vector

$$\underset{\sim}{S} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix} \tag{27}$$

This vector will be referred to as the input vector.

The position of the relays of all of the minterm recognition units can be expressed in the form of a connection matrix

$$C = \begin{bmatrix} c_{11} & c_{12} \cdots & c_{1j} \cdots & c_{1n} \\ c_{21} & c_{22} \cdots & c_{2j} \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{i1} & c_{i2} \cdots & c_{ij} \cdots & c_{in} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{m2} \cdots & c_{mj} \cdots & c_{mn} \end{bmatrix} \tag{28}$$

where c_{ij} is the connection of the summing device of the ith minterm recognition unit to the jth sensor unit or inverter. The value of c_{ij} will be zero if the connection

is made to the inverter and one if the connection is made to the sensor unit. Each row vector is thus identical to the corresponding model vector.

The complement of a binary matrix or vector is an array in which each of the elements of the original array is replaced by its complement. Thus

$$\underset{\sim}{\overline{S}} = \begin{bmatrix} \overline{s}_1 \\ \overline{s}_2 \\ \vdots \\ \overline{s}_n \end{bmatrix} \tag{29}$$

and

$$\overline{C} = \begin{bmatrix} \overline{c}_{11} & \overline{c}_{12} \cdots & \overline{c}_{1n} \\ \vdots & \vdots & \vdots \\ \overline{c}_{m1} & \overline{c}_{m2} \cdots & \overline{c}_{mn} \end{bmatrix} \tag{30}$$

The output of the summing device of each minterm recognition unit is the degree of correlation of the input vector with the minterm recognition unit model vector. The degrees of correlation of all of the minterm recognition units can be arranged in a column vector

$$\underset{\sim}{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} \tag{31}$$

where each d_i is the degree of correlation of the input vector with the ith minterm recognition unit model vector. This vector can be calculated from

$$\underset{\sim}{d} = C \underset{\sim}{S} + \overline{C}\,\overline{\underset{\sim}{S}} \tag{32}$$

The summing device outputs of all m of the minterm units are the inputs for the MAX gate of the control unit (Fig. 8b). The MAX gate has as its output the maximum value of its inputs. A simple diode OR gate will produce an output that is the maximum value of its inputs and can be used as a MAX gate.

The output of the MAX gate is the value of the highest degree of correlation of the input vectors to any of the model vectors. This can be expressed as

$$D = \max \left| d_i \right| \tag{33}$$

where D is the maximum degree of correlation.

The algebraic sign of D is then changed by a polarity changer. It is possible that the polarity changing operation could be accomplished by reversing two leads.

The output of the polarity changer is connected to the inputs of the threshold gates of all of the minterm recognition units. The output of the summing device of a minterm recognition unit is also an input for that unit's threshold gate. The value of the threshold value is zero,

and all input weights are one. The threshold gate will have a logical one output if the output of the summing device is equal to the maximum degree of correlation. This can be written as

$$Mi = \begin{cases} 1 \text{ if } d_i - D = 0 \\ 0 \text{ if } d_i - D < 0 \end{cases} \tag{34}$$

where Mi is the output of the ith minterm recognition unit. Because D is the maximum degree of correlation, d_i can never be greater than D, but several minterm recognition units may have degrees of correlation equal to the maximum value. Thus, the input vectors will always be placed in the same classifications as the model vectors to which they have the highest degree of correlation.

An OR gate is used to combine the outputs of all of the minterm recognition units associated with model input vectors that are to be placed in the same classification. Thus, a logical one OR gate response indicates that the input vector has a maximum degree of correlation with one of the model vectors of the OR gate's class.

Encoding Networks. The response is in a decoded form. It can be encoded by using a coding network to connect the OR gate outputs to at least $\log_2 k$ terminals, where k is the number of classes.

A more flexible way to provide a coded output would

be to connect each minterm recognition unit to all of the OR gates through switches. In this case a minimum of $\log_2 k$ OR gates is required instead of k OR gates. The switches can be opened or closed during the training process so that any desired set of OR gate responses can be obtained for the output of any minterm recognition unit. Thus, the output can be put in coded form.

Error Detecting. It is possible that a pattern will be equally close to two or more model vectors. In this case, more than one minterm recognition unit would respond to the pattern. If these minterm recognition units are connected to different OR gates, the pattern will be classified in two different classes. If all of the OR gate outputs are connected to a threshold gate with a threshold value of two and unity input weights, the threshold gate will give an error signal when two or more OR gates have a logical one output. If the encoded output is used, the threshold gate must be connected to the minterm recognition unit outputs.

Another type of error occurs when a pattern is classified that does not have a high degree of correlation with any of the model vectors. This type of pattern probably does not belong in any of the classes and should be rejected. A reject signal can be obtained by connecting a threshold gate to the output of the polarity changer. If

-D, the negative of the maximum degree of correlation, exceeds some negative threshold value a reject signal would appear at the output of the gate. A good value for the magnitude of the threshold probably is the average of the maximum degrees of correlation of each model input vector with the other model vectors. This value was used during the computer simulation experiments to be discussed later.

Possible Modifications. Several possible modifications can be made to the system. The relays may be replaced by electronic components for a more sophisticated device. If a less complicated unit is desired, the relays may be replaced by toggle switches to make an organizable network.

An electronic version of a minterm recognition unit is shown in Fig. 9. The relays have been replaced by bistable multivibrators and AND gates. Before each minterm recognition unit is trained to recognize a model vector, all of the bistable multivibrators are set to their zero states by a pulse on the Reset lead. An enabling signal L_i allows the AND gate on the input of the bistable multivibrator to have a logical one response if the corresponding sensor unit is excited when a model pattern is presented. Thus, a multivibrator will switch to the one state only if the sensor unit is excited.

After the minterm recognition unit has been organized,

the enabling signal L_i is turned off. During the pattern recognition process, one and only one of the AND gates connected to the outputs of the ith multivibrator will have a logical one response if the ith term of the input vector equals the ith term of the model vector. Otherwise, the outputs of both of the ith AND gates are zero. These AND gate outputs are added in the summing device to determine the degree of correlation. The mathematical analysis of the pattern recognition process for the electronic version is the same as the analysis for the relay version of the minterm recognition unit.

The pattern recognition capabilities of the SOPRU may possibly be improved by using some of the techniques discussed in Chapter II. The minterm recognition unit input weights could be adjusted to emphasize the important input vector components. This would have to be done with the constraint that the sum of the weights must always be a constant equal to n so that the model vectors will still be classified correctly. More study is necessary to determine the weight adjustment procedure.

Computer Simulation of a SOPRU

A computer program (Appendix C) was written by the author to simulate a 25-input SOPRU with 25 minterm recognition units so that the noisy pattern recognition capabilities of the SOPRU could be investigated experimentally.

The number of classes equaled the number of model input vectors used, so no OR gates were simulated. The first portion of the program automatically determined the average of the maximum degrees of correlation of each model input vector with the other model vectors. This value was used as a threshold to determine if a reject message should be printed. Only one problem was simulated.

Problem. The same data that was used in Problem 2 of Chapter II was used in this problem so that a comparison of the results could be made. The 25 input vectors of Eqs (11) will not be repeated here.

The model vectors were used as the connection matrix. Thus, there was no learning process necessary for this problem. The 625 one-digit-noisy input vectors were presented to the SOPRU to see how each was classified. Then 600 vectors with two noisy digits were shown to the SOPRU. Thus, 1225 samples were given to the machine.

Problem Results. The average value of the maximum degrees of correlation of each model vector with the other model vectors was calculated by the computer to be 20. None of the vectors presented to the SOPRU had maximum degrees of correlation below that threshold value, so no reject messages were printed.

Twelve of the 625 vectors with one noisy digit were placed in two different classifications. Fifteen of the

600 vectors with two noisy digits were also given dual classifications. In all 1225 cases one of the classes in which the pattern was classified was the correct class. Thus, the one-digit-noisy patterns were classified without ambiguity for slightly more than 98% of the samples, and the vectors with two noisy digits were given singular classifications for slightly less than 98% of the samples. Fig. 10 shows the noisy characters that were given dual classifications. If these patterns are compared to the model patterns of Fig. 5, it can be seen that the noisy patterns are equally close to the two model patterns listed under the noisy patterns.

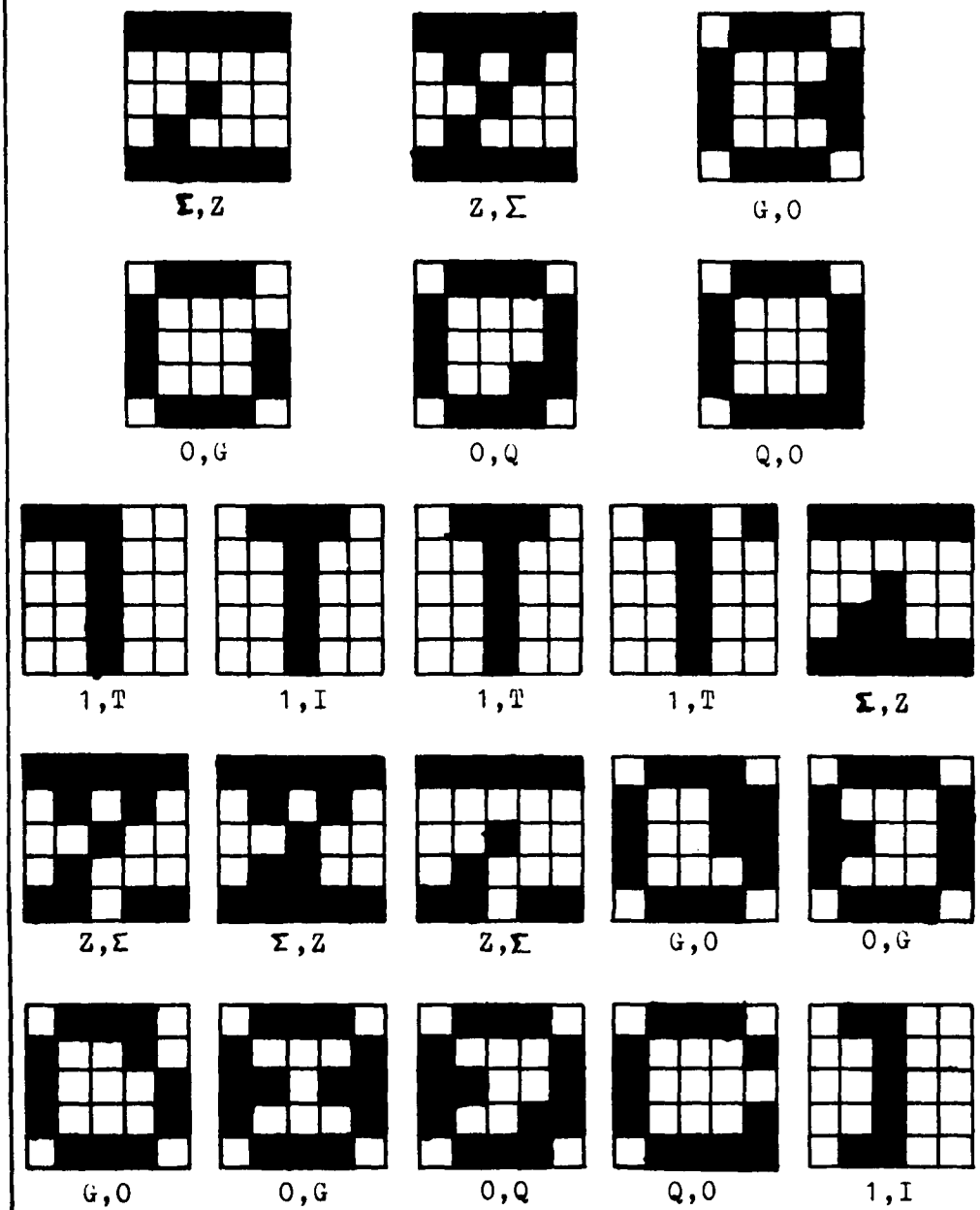


Fig. 10

Noisy Patterns Given Dual Classification by SOPRU

V. Results and Conclusions

Some of the advantages and disadvantages of the three classes of self-organizing machines discussed in the previous chapters can be seen by comparing the number of components required and the results of the computer simulation problems. These comparisons will be made, some conclusions will be drawn, and several recommendations for further study will be stated.

Comparison of Results

Number of Units Required. Because all of the machines discussed require essentially the same number of input and output units, no comparison of these numbers needs to be made. There is a distinct difference in the number of intermediate logical devices necessary, however.

The perceptron requires a maximum of 2^n association units, where n is the number of inputs. The minimum number is a function of the number of model vectors the perceptron is required to learn, the learning speed desired, and the desired noisy pattern classification ability. In general, this minimum number can be much less than 2^n.

A completely general Self-Organizing Binary Logical Network must have at least 2^n AND gates and 2^n statistical switches. If all of the model vectors that the machine

will be required to learn are known in advance, fewer AND gates and statistical switches may be used, but the machine will not be able to learn all of the possible patterns.

The Self-Organizing Pattern Recognition Unit requires only as many minterm recognition units as there are model input vectors.

Learning Time. The table below shows the number of times the machines had to see each model input vector during the organization processes.

	Problem 2 Learning Time	Problem 3 Learning Time
Perceptron	32	1
SOBLN	-	6
SOPRU	1	-

A slight modification of the program would have allowed the SOBLN to learn the model vectors after seeing each vector only once.

Noisy Pattern Recognition Levels. The abilities of the machines to recognize the noisy patterns of Problems 2 and 3 are listed in the table below. The recognition level is the fraction of the patterns that were recognized correctly.

	Problem 2 Recognition Levels		Problem 3 Recognition Levels
	One Digit Noisy	Two Digits Noisy	One Digit Noisy
Perceptron	85%	75%	100% for 4 perceptrons 94% for 2 perceptrons
SOBLN	—	—	51%
SOPRU	98%	98%	—

It should be recalled that the SOPRU was required to separate the vectors into 25 different classes for problem 2, but the perceptron had to distinguish between two classes only. The perceptron, however, was handicapped because it had a small ratio of association units to sensor units, and it only had one response unit.

Conclusions

The conclusions, based on this investigation are as follows:

1. Self-organizing machines which use adjustable-weight threshold logic, such as perceptrons with high ratios of association units to sensor units, are well suited to solve pattern recognition problems because of their ability to classify most noisy patterns correctly. Their major disadvantage is the relatively large number of times the model input vectors must be presented.

2. Statistical-switching machines, such as the Self-Organizing Binary Logical Network, cannot be used to recognize noisy patterns. Furthermore, they are not practical for problems requiring large numbers of inputs (sensor units) because they require 2^n AND gates and 2^n statistical switches. The major advantage of the SOBLN is the relatively few number of times the model input vector must be shown to the machine.

3. Machines using correlation techniques, such as the self-organizing pattern recognition unit, combine some of the advantages of the perceptron and the SOBLN. They have a high degree of ability in noisy pattern recognition. They require relatively few components. Furthermore, the SOPRU must see each model vector only once.

Recommendations for Further Study

Several topics are recommended for further investigation:

1. It is possible that the set of weights determined by the α-error correction procedure in the perceptron is not the optimum feasible solution for noisy pattern recognition. It may be possible to apply some of the techniques of linear programming to determine an optimum feasible solution that would maximize the distance of each model vector from the separating hyperplane. This may correspond to maximizing slack variables.

2. The weights on the inputs of the SOPRU all have a value of one. An adjustment procedure could be derived as suggested in Chapter IV that would improve the noisy pattern recognition capabilities of the SOPRU by changing the value of these weights.

3. An organizable pattern recognition unit could be constructed using the SOPRU recognition techniques. The relays could be replaced by inexpensive toggle switches. The summing devices could be built using Kirchhoff adders.

Bibliography

1. Brown, F. M., and L. E. Knapp. Basic-Unit Operation in the Steele Neuron. Bionics Memorandum. Wright-Patterson Air Force Base: Electronic Technology Laboratory, Wright Air Development Division, March 1961.

2. Buroker, Harold E. Noise Sensitivity in Probability State Variable Devices. Master's Thesis. Wright-Patterson Air Force Base: Air Force Institute of Technology, August 1963.

3. Carne, E. B., et al. A Study of Generalized Machine Learning. Aeronautical Systems Division Technical Documentary Report ASD-TDR-62-166. Wright-Patterson Air Force Base: Electronic Technology Laboratory, Aeronautical Systems Division, April 1962.

4. Choisser, John P., and John W. Sammon. "A New Concept in Artificial Intelligence." National Aviation Electronics Conference Proceedings (1963). Dayton, Ohio: National Aviation Electronics Conference, May 1963, pp. 340-347.

5. Fischler, Martin A. "Hyperplane Techniques in Pattern Recognition." Proceedings of the IEEE, 51:497-498 (March 1963).

6. Highleyman, W. H. "An Analog Method for Character Recognition." IRE Transactions on Electronic Computers, EC-10: 502-512 (September 1961).

7. -----. "Linear Decision Functions with Application to Pattern Recognition." Proceedings of the IRE, 50:1501-1514 (June 1962).

8. Mattson, R. L. A Self-Organizing Binary System. Technical Report LMSD-288029. Sunnyvale, California: Lockhead Aircraft Corporation, September 1959.

9. Murray, Albert E. Perceptron Conference Discussion. Buffalo, New York: Cornell Aeronautical Laboratory, Inc., June 1960.

10. The Perceptron. Published Speech. Buffalo, New York: Cornell Aeronautical Laboratory, Inc., December 1960.

11. Rosenblatt, Frank. On the Convergence of Reinforcement Procedures in Simple Perceptrons. Cornell Aeronautical Laboratory Report No. VG-1196-G-4. Buffalo, New York: Cornell Aeronautical Laboratory, Inc., February 1960.

12. Taussky, Olga, and John Todd. "Generation and Testing of Pseudo-Random Numbers," in Symposium on Monte Carlo Methods, edited by Herbert A. Meyer. New York: John Wiley and Sons, Inc., 1956, pp. 15-18.

Appendix A

Computer Simulation Program for a Simple Perceptron

A computer program to simulate a simple perceptron on an IBM 1620 digital computer is shown on page 71. A flow chart for this program is shown in Fig. A-1. The number of sensor (S) units for the perceptron simulated is twenty-five, and the number of association (A) units is fifty. Only one response (R) unit is used.

The outputs of the A units can be stored for a maximum of twenty-five different input vectors. Thus, during the organization process, the outputs of the A units do not have to be recalculated after all of the input vectors have been presented to the computer. After the response for each input vector has been calculated, the computer will type out the number of the input vector, the actual response, and the desired response. If console switch 3 is on, the weights will be adjusted using the α-error correction procedure.

The program was written in the AFIT Fortran processing language. AFIT Fortran allows a free-style input format on the data cards. Thus, the input data may be punched on cards in any format with spaces or commas between each number. A memory capacity of 40,000 decimal digits is required

to contain the program and the data.

Several simple modifications of this program can be made to adapt it to special problems. The final value of "DO" statements 31, 5, and 9 can be changed to any number less than twenty-five to change the number of sensor (input) units. If the number is made higher than twenty-five, the appropriate dimension statements must be changed. Similarly, the final values of "DO" statements 11, 12, 32, 13, and 19 can be changed to vary the number of A units. The dimension statements must also be changed if the number of A units desired is more than fifty.

A table of console switch settings with their functions is given below.

	ON	OFF
SW#1	Read new input vectors	Do not read input vectors
SW#2	Calculate A unit response	Use stored A unit response
SW#3	Adjust weights on R unit inputs	Do not adjust weights on R unit inputs

Fig. A-1

Flow Diagram for Perceptron Simulation Program

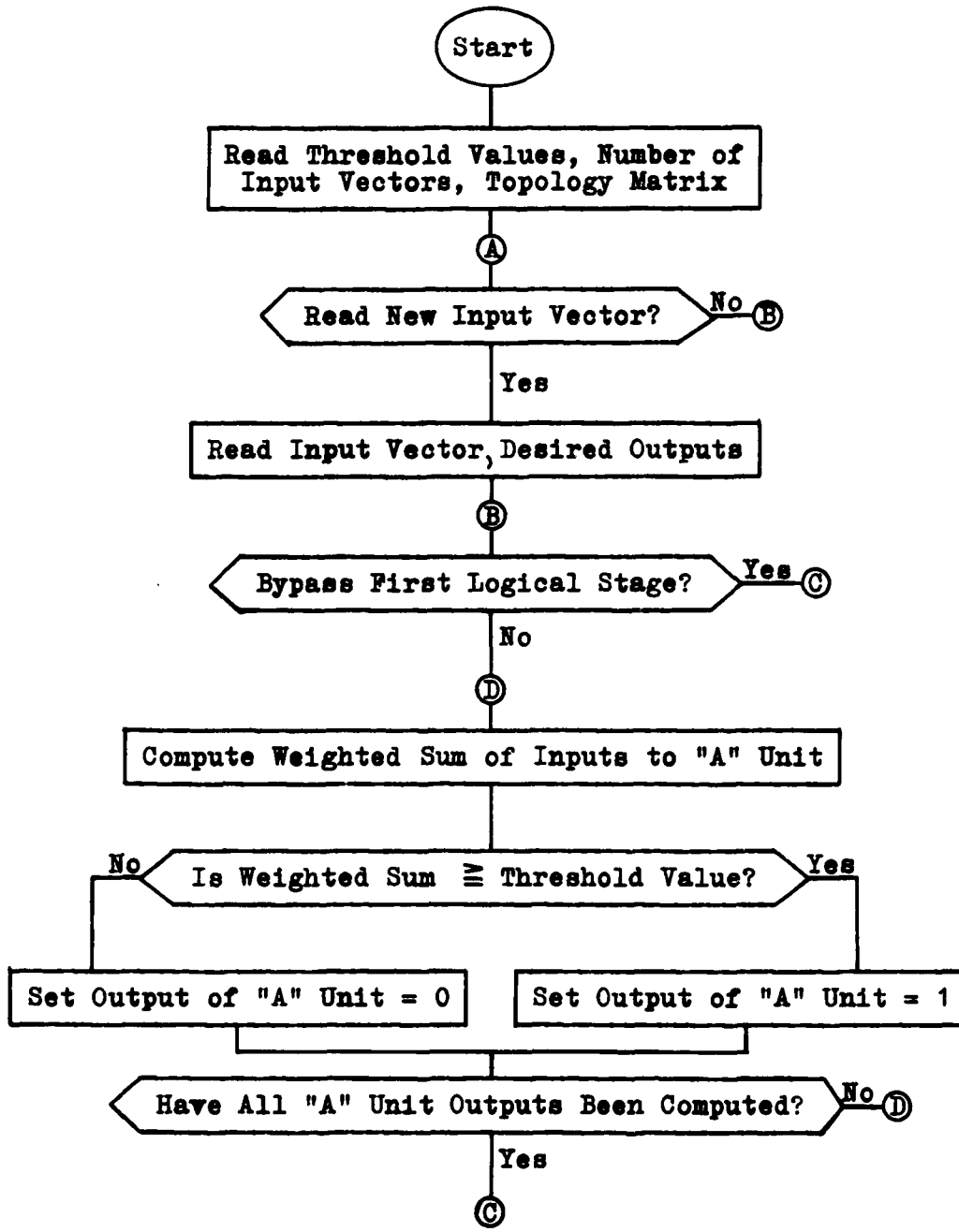
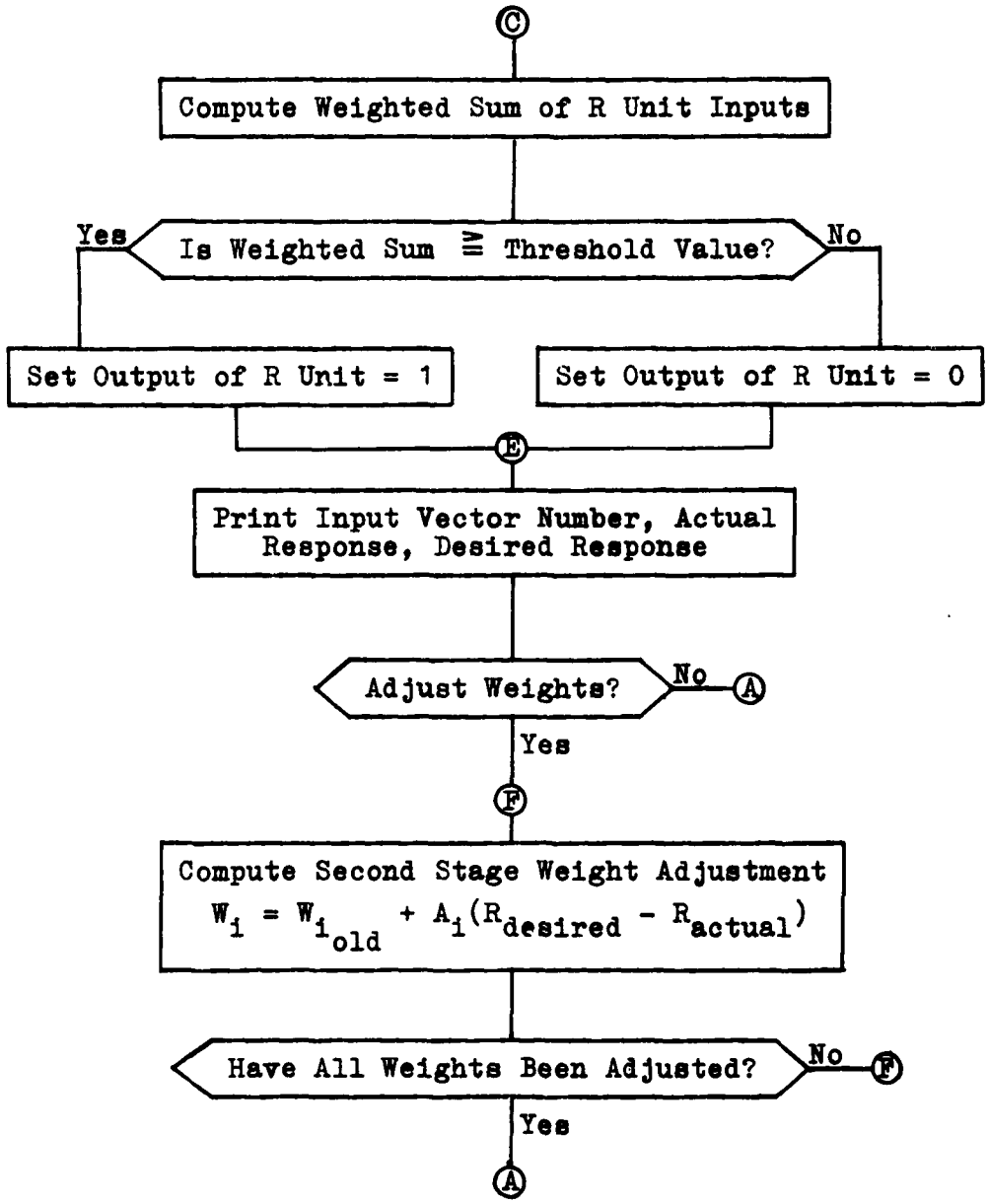


Fig. A-1

Flow Diagram for Perceptron Simulation Program



Computer Program for a Simple Perceptron

```
C   PERCEPTRON PROGRAM
    DIMENSION IFETA(50),IW(10),IT(25,50),IS(25)
    DIMENSION IRD(25),IA(25,50),IP(50)

1   READ,KFETA,M                    Read R unit threshold value
                                    & number of input vectors

11  DO 2 I=1,50                     Read second stage initial
2   READ,IW(1)                      weights

12  DO 3 I=1,50
3   READ,IFETA(1)                   Read A unit threshold value

31  DO 4 I=1,25
32  DO 4 J=1,50                     Read first stage topology
4   READ,IT(I,J)                    matrix

41  PRINT 26                        Print column heading
    DO 22 K=1,M                     Set vector number counter
    IF(SENSE SWITCH 1)5,7           Bypass input vector read

5   DO 6 J=1,25
6   READ,IS(J)                      Read input vector

    READ,IRD(K)                     Read desired response

7   IQ=0
13  DO 17 I=1,50
    IF(SENSE SWITCH 2)8,15          Compute weighted sum of A
8   IP(I)=0                         unit inputs
9   DO 10 J=1,25
10  IP(I)=1P(1)+1T(1,J)*1S(J)

15  IA(KI)=0
    IF(IP(I)-IFETA(I))17,16,16      Determine A unit response
16  IA(K,I)=1

17  IQ=IQ+IW(I)*IA(K,I)             Compute weighted sum of
                                    R unit

    IR=0
    IF(IQ-KFETA)42,18,18            Determine R unit response
18  IR=1
```

```
42  PRINT 25,K,IR,IRD(K)          Print results
    IF(SENSE SWITCH 3)19,22       Bypass weight adjustment

19  DO 20 I=1,50
20  IW(I)=IW(I)+IA(K,I)*(IRD(K)-IR)   Adjust second stage
22  CONTINUE                          weights

    GO TO 41
25  FORMAT(/I3,I2,I2)
26  FORMAT(//8H SN R RD)
    END
```

Symbol Table

IA(K,I) = response of ith A unit to Kth input vector

IFETA(I) = threshold value of ith A unit threshold gate

IP(I) = weighted sum of inputs to ith A unit

IQ = weighted sum of inputs to R unit

IR = response of R unit

IRD(K) = desired response of R unit to Kth input vector

IS(J) = value of jth bit of binary input (sensor) vector

IT(I,J) = weight value of input to the ith A unit from the jth S unit

IW(I) = weight value of input to R unit from the ith A unit

I = row subscript

J = column subscript

K = input vector number

KFETA = threshold value of R unit threshold gate

M = number of input vectors

Appendix B

Computer Simulation Program for a Self-Organizing Binary Logical Network

The computer program shown on page 79 will simulate a nine-input Self-Organizing Binary Logical Network. For a completely general pattern recognition device, a minimum of one statistical switch and one AND gate is required for each minterm; therefore, 2^9 statistical switches and AND gates were simulated.

The program was written in the Symbolic Programming System (SPS) language for an IBM 1620 computer with indirect addressing. SPS was used because of the difficulty of generating a random number and because of the large memory capacity required when FORTRAN is used for a simulation program of this sort.

The program works in the following manner. The first stage connection matrix consisting of all of the binary numbers from 0 to 511 is read into memory. Each binary number corresponds to the minterm associated with one of the AND gates. The model input vectors with their desired responses are read into memory in binary form. Each input vector is compared to each binary number of the connection matrix. The binary connection vector that is equal to the

input vector corresponds to the AND gate which will have a logical one output. The outputs of all other And gates are zero. A random number is then generated using Lehmer's method (Ref 12:17). The random number is compared to a threshold value corresponding to the counter value of the statistical switch associated with the AND gate which had a logical one output. If the random number is less than the threshold value, a logical one is the output of the OR gate. Otherwise, the output is a logical zero. During the training period, the statistical switch counter is increased to a maximum of twelve if the desired OR gate output is a logical one, and the counter is decreased to a minimum of zero if the desired output is zero. Thus, the training process is complete after the response for each input vector has been calculated six times. A flow chart for the program is shown in Fig. B-1.

The input data must be punched on cards using a special format. There must be a flag over the left-hand digit of each number, and there must be no spaces between digits. The initial random number must be punched in the first eight columns of the card. Eight 9-digit binary numbers must be punched on each of 64 cards for the connection matrix. Each input vector must be punched in the first nine columns of a card. All of the desired responses must be punched in the first columns of one card with a zero after

each desired response.

During the training period, switch 1 may be turned off to speed up the training operation by bypassing the first stage logic after the AND gate responses for each model input vector have been determined. Switch 2 controls the reward and punishment of the statistical switches and should be turned off when training is completed. Switch 3 allows new input vectors to be presented to the machine. A table of the console switch settings is shown below.

SWITCH	ON	OFF
SW#1	Calculate AND gate responses	Bypass AND gate response calculation
SW#2	Adjust statistical-switch probability	Bypass probability adjustment
SW#3	Read new input vectors	Do not read new input vectors

Fig. B-1

Flow Chart for a Computer Simulation of a Self-Organizing Binary Logical Network

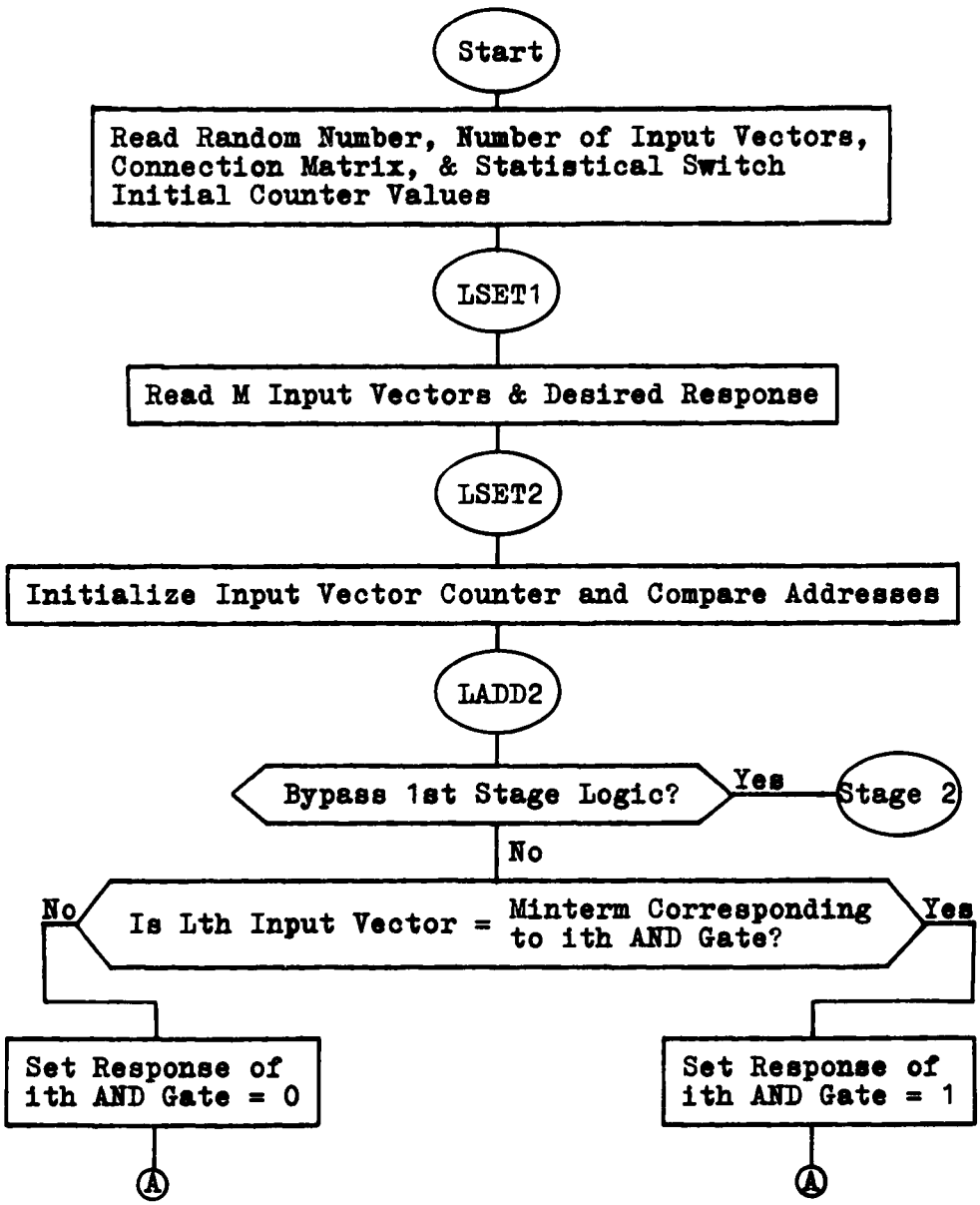


Fig. B-1

Flow Chart for a Computer Simulation of a Self-Organizing Binary Logical Network

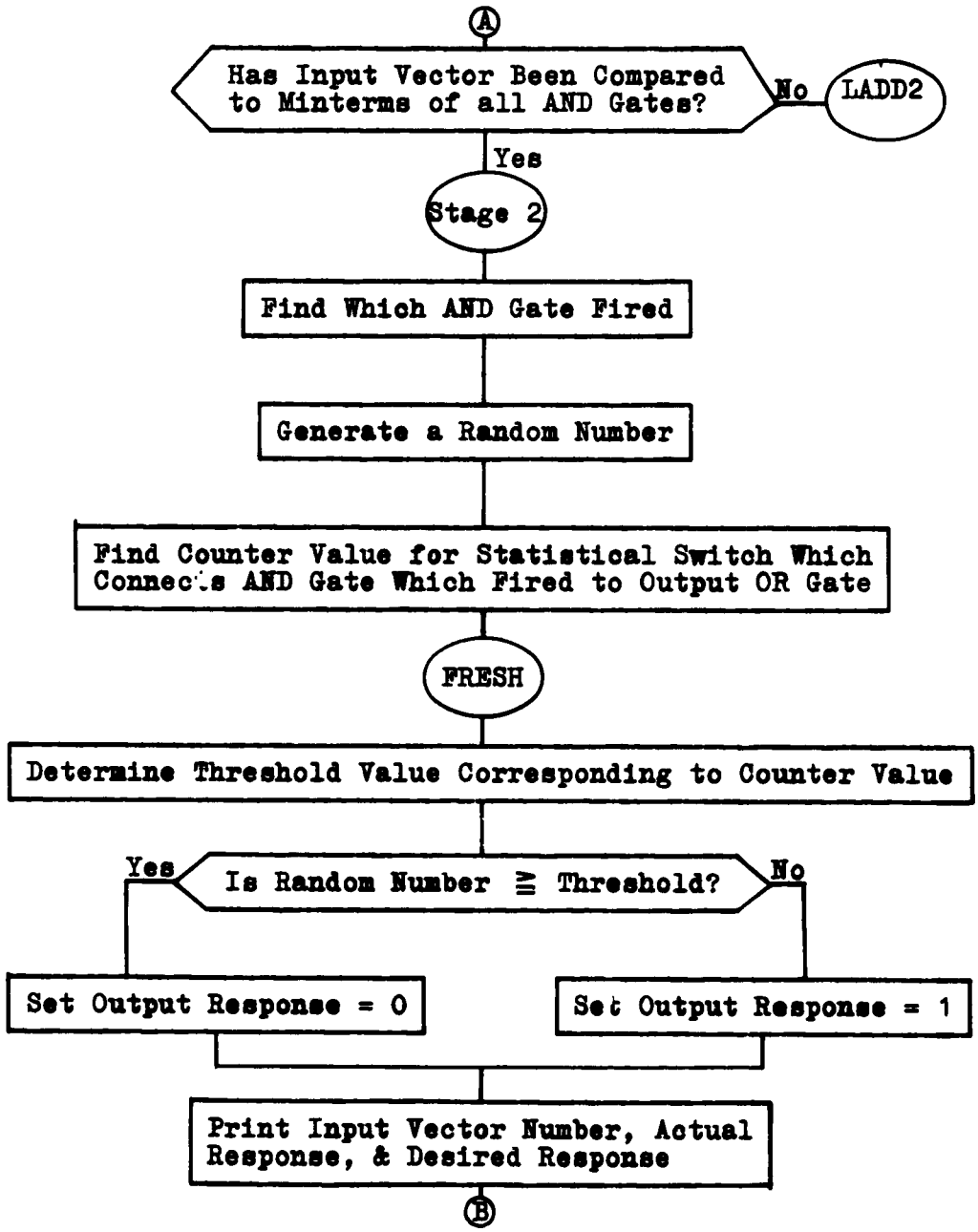
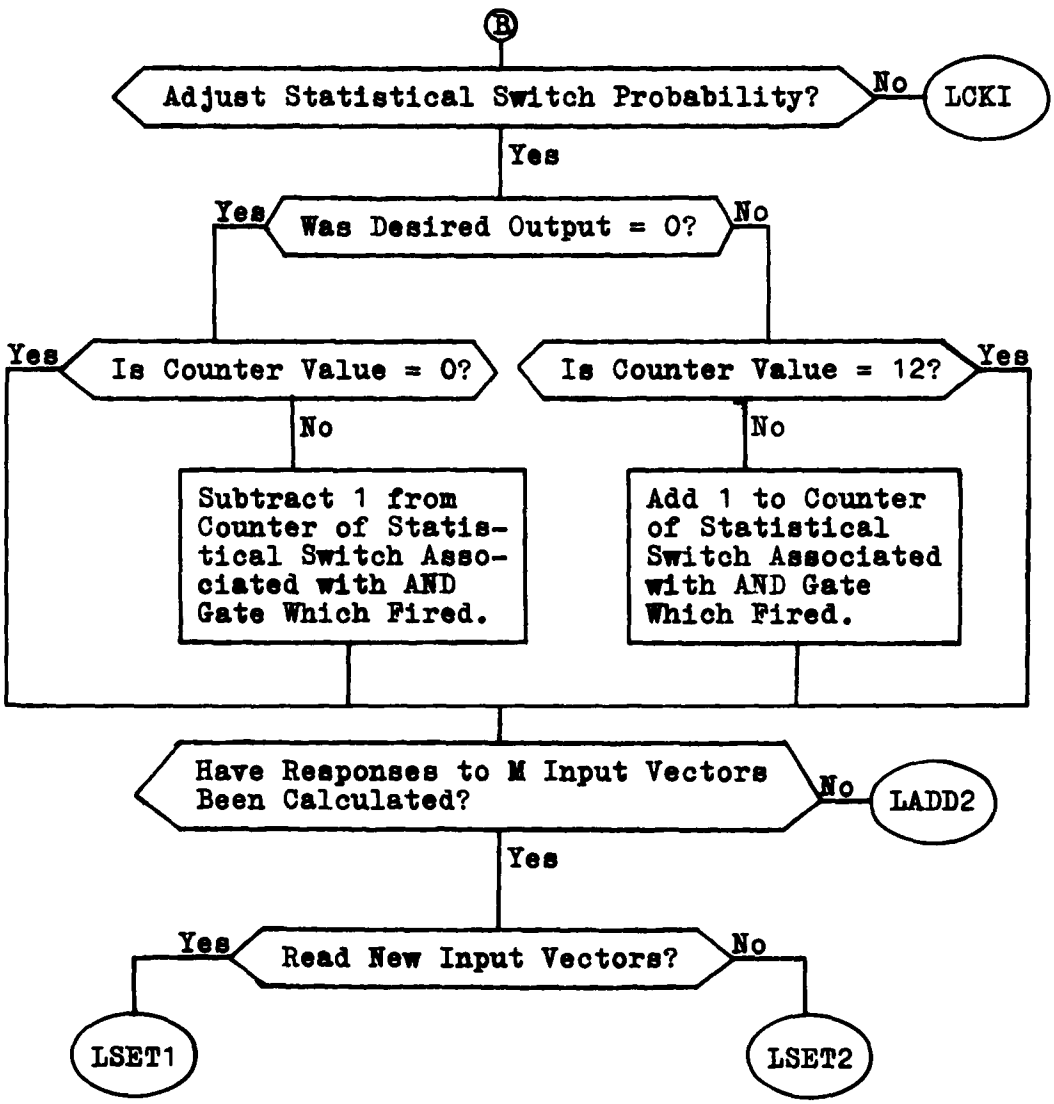


Fig. B-1

Flow Chart for a Computer Simulation of a Self-Organizing Binary Logical Network



Self-Organizing Binary Logical Network Program

```
START   RNCD   RN-7                 Read random number
                                    Read number of input vectors
        TFM    N,0,10
        TFM    ADDR,CIJ-8
RCN     AM     N,1,10               Read 1st stage connection
        RNCD   -ADDR                matrix
        AM     ADDR,72
        CM     N,64,10
        BL     RCN

        TFM    N,0,10
        TFM    ADDR,KI-1
RKN     AM     N,1,10               Read statistical-switch
        RNCD   -ADDR                counter initial values
        AM     ADDR,64
        CM     N,16,10
        BL     RKN

LSET1   TFM    L,0,10
        TFM    RES+6,SLI-8
LADD1   AM     L,1,10
RES     RNCD   SLI-8                Read input vectors
        AM     RES+6,9
        C      L,2
        BL     LADD1

        RNCD   IRDL                 Read desired response

LSET2   TFM    L,0,10               Reset counter & addresses
        TFM    COMP+11,SLI-9

LADD2   AM     L,1,10
        MM     L,512,9              Calculate address of re-
        TF     ADDR,99              sponse for 1st AND gate
        AM     ADDR,RI-512

        RNC1   STAGE2               Bypass to stage 2

        TFM    COMP+6,CIJ
        AM     COMP+11,9
        TFM    I,0,9                Compare input vector to minterm
IADD2   AM     I,1,9                of ith AND gate
COMP    C      CIJ,SLI
        BZ     RSET1
```

```
RSET0   TDM   -ADDR,0          Set response = 0

        AM    ADDR,1
        AM    COMP+6,9        Modify addresses

        CM    I,512,9
        BL    IADD2           Checked all minterms?
        B     STAGE2

RSET1   TDM   -ADDR,1
        B     RSET0+12        Set response = 1

STAGE2  MM    L,512,9
        TF    ADDR,99
        AM    ADDR,RI-512
        TFM   I,0,9
IADD7   AM    I,1,9           Find which AND gate fired
DIG     BD    RAND,-ADDR
        AM    ADDR,1
        B     IADD7

RAND    MM    RN,23,10
        SF    92
        S     99,91           Generate random number
        TF    RN,99

        MM    I,2,10
        AM    99,KI-2         Find statistical switch
        TF    ADDR1,99        counter address

FRESH   MM    -ADDR1,8,9      Find address of threshold
        AM    99,THRES

        C     RN,-99          Compare random number to
        BH    RO              threshold

        TDM   R,1
        B     PRINT           Set response = 1

RO      TDM   R,0             Set response = 0
PRINT   RCTY

        BD    TYP,L-1
        WNTY  L
        B     SPACE           Print input vector
TYP     WNTY  L-1
```

```
SPACE   SPTY
        WNTY    R               Print actual response
        SPTY

        MM      L,2,9
        TF      ADDR2,99
        AM      ADDR2,IRDL-1
        TD      -ADDR2,400      Print desired response
        SM      ADDR2,1
        WNTY    -ADDR2

        BNC2    LCK1            Bypass counter adjustment
        BD      ADDK1,-ADDR2    Branch if desired output = 1
        BD      SUBK1,-ADDR1    Branch if counter value = 0
        B       LCK1

ADDK1   CM      -ADDR1,12,10
        BNL     LCK1            Branch if counter value = 12

        AM      -ADDR1,1,10
        B       LCK1            Add 1 to counter value

SUBK1   SM      -ADDR1,1,10     Subtract 1 from counter value

LCK1    C       L,2             Have all responses been
        BL      LADD2           determined?

        BC3     LSET1           Read new input vectors
        B       LSET2           Do not read new input vectors

THRES   DC      8,00000000
        DC      8,01000000
        DC      8,03000000
        DC      8,06000000
        DC      8,12000000
        DC      8,25000000
        DC      8,50000000      Statistical switch threshold
        DC      8,75000000      values
        DC      8,87000000
        DC      8,93000000
        DC      8,96000000
        DC      8,98000000
        DC      8,99000000

N       DS      2               Card counter
L       DS      2               Input vector subscript
        DC      1,0
I       DS      3               Row subscript
RI      DSS     5120            Response of ith AND gate
```

```
ADDR    DS    5        Address storage for R_i   }
ADDR1   DS    5        Address storage for K_i   } For indirect addressing
ADDR2   DS    5        Address storage for RD_1  }

R       DS    2        Or gate response
        DC    1,@
RN      DS    8        Random Number
CIJ     DSB   9,512    Minterm of ith AND gate
KI      DSB   2,512    ith statistical switch counter
SLI     DSB   9,10     lth input vector
IRDL    DSS   20       Desired response of lth input vector
        DEND  START
```

Appendix C

Computer Simulation Program for a Self-Organizing Pattern Recognition Unit

The computer program for the simulation of a Self-Organizing Pattern Recognition Unit (SOPRU) was written in AFIT FORTRAN for an IBM 1620 computer. The program will simulate a 25 input SOPRU with 25 minterm recognition units. Since a minimum of one minterm recognition unit is required for each class of patterns to be recognized, the SOPRU simulated will separate a maximum of 25 different pattern groups.

The first step in the program is to read into memory the model input vectors. If console switch 1 is on, a threshold value is then determined by finding the maximum degree of correlation of each model vector with the other model vectors and then finding the average of these maximum correlation values. If the threshold value is already known, the threshold value calculation can be bypassed by turning console switch 1 off and typing in the threshold value.

Twenty-five input vectors representing patterns to be recognized are then read into memory. Each digit of a binary input vector is then compared to the corresponding

digit of a model vector. If the digits are equal, a one is added to an accumulator. After all 25 digits have been checked, the number in the accumulator, which is the degree of correlation, is compared to previously determined degrees of correlation of the input vector to the other model vectors to see which is a maximum. The maximum value of these degrees of correlation is then compared to the degree of correlation of the input vector with each model vector. Each model vector with a degree of correlation to the input vector equal to the maximum degree of correlation corresponds to a minterm recognition unit with an output equal to a logical one.

If the input vector was improperly classified, the input vector number, the number of the minterm recognition unit with a logical one output, and the maximum degree of correlation are typed. If the maximum correlation is less than the previously determined threshold, the word (Reject" is typed. After 625 patterns have been classified, the program stops.

A flow chart is shown in Fig. C-1, followed by the program.

Fig. C-1

Flow Chart for a Computer Simulation of a Self-Organizing Pattern Recognition Unit

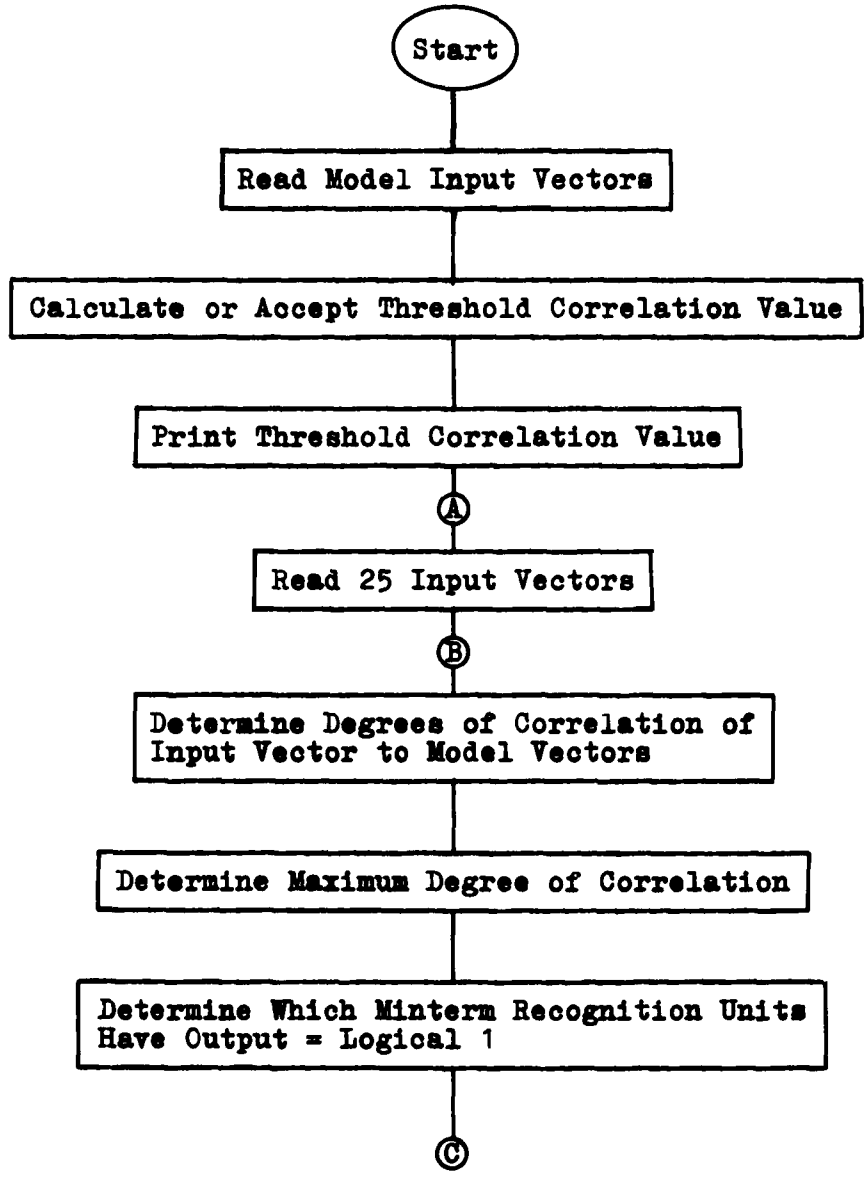
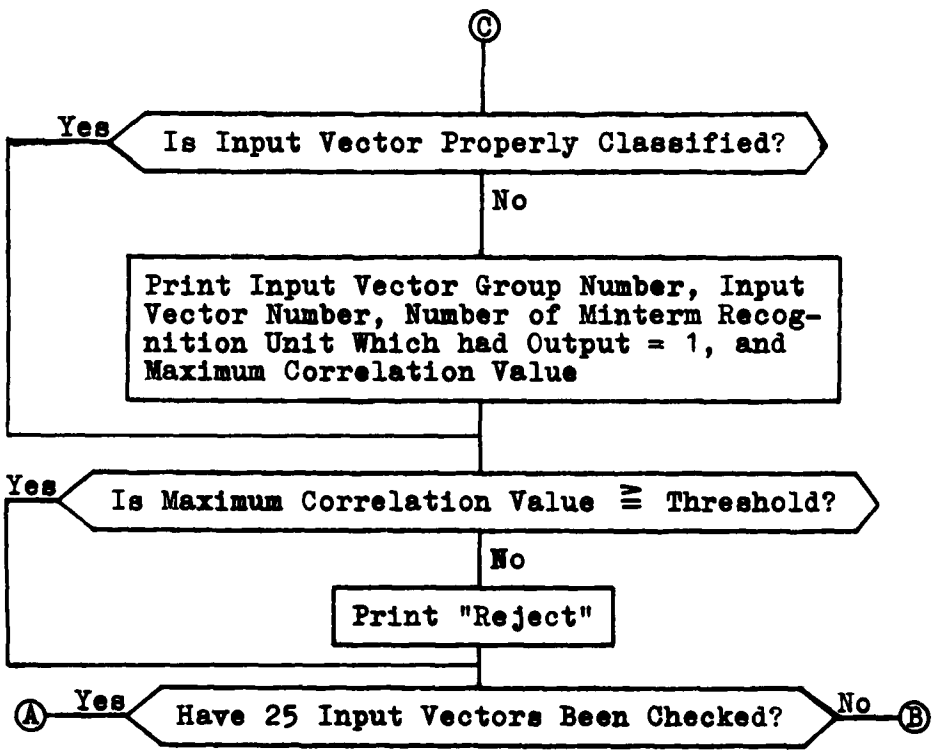


Fig. C-1

Flow Chart for a Computer Simulation of a Self-Organizing Pattern Recognition Unit



Computer Program for Self-Organizing Pattern Recognition Unit

```
C    SOPRU 2
     DIMENSION MOD(25,25),IFRES(25)
     DIMENSION ISUM(25),INPUT(25,25)

     DO 1 L=1,25                              Read model input
     DO 1 I=1,25                              vectors
1    READ 25,MOD(L,I)

     IF(SENSE SWITCH 1)22,23                  Bypass minimum correla-
                                              tion calculation
22   IFETA=0
     DO 6 M=1,25
     IFRES(M)=0
     DO 5 L=1,25
     IF(L-M)21,5,21
21   ISUM(L)=0
     DO 3 I=1,25
     IF(MOD(M,I)-MOD(L,I))3,2,3               Calculate minimum
2    ISUM(L)=ISUM(L)+1                        correlation value
3    CONTINUE
     IF(ISUM(L)-IFRES(M))5,5,4
4    IFRES(M)=ISUM(L)
5    CONTINUE
6    IFETA=IFETA+IFRES(M)
     IFETA=IFETA/25
     GO TO 7

23   ACCEPT,IFETA                             Read minimum
                                              correlation value
7    PRINT 26,IFETA                           Print minimum
                                              correlation value
     DO 18 M=1,25                             Set group counter

     DO 8 K=1,25
     DO 8 I=1,25                              Read 25 input vectors
8    READ 25,INPUT(K,I)

     DO 18 K=1,25                             Set input vector
                                              counter
     JSUM=0
```

```
    DO 12 L=1,25                          Set minterm counter

    ISUM(L)=0
    DO I=1,25
    IF(INPUT(K,I)-MOD(L,I))10,9,10        Determine degree of
9   ISUM(L)=ISUM(L)+1                     correlation
10  CONTINUE

    IF(ISUM(L)-JSUM)12,12,11
11  JSUM=ISUM(L)                          Determine maximum
12  CONTINUE                              correlation value

    DO 18 L=1,25                          Determine which minterm
    IF(ISUM(L)-JSUM)18,13,13              recognition units fired

13  IF(L-K)14,15,14                       If vector is not clas-
14  PRINT 27,M,K,L,ISUM(L)                sified correctly, print
                                          results

15  IF(JSUM-IFETA)16,18,18                If correlation value is
16  PRINT 28,M,K,L,ISUM(L)                below minimum correla-
18  CONTINUE                              tion value, print "Reject"

    STOP
25  FORMAT(I50,30X)
26  FORMAT(/20HMINIMUM CORRELATION=,I3)
27  FORMAT(/I3,I4,I4,I6)
28  FORMAT(/6HREJECT,I4,I4,I4,I4,)
    END
```

Symbol Table

I = row subscript

IFETA = threshold (minimum correlation) value

IFRES(M) = degree of correlation of mth model vector with other model vectors

INPUT(K,I) = ith digit of kth input vector

ISUM(L) = degree of correlation of lth model vector with input vector

JSUM = maximum degree of correlation

K = input vector number

L = minterm recognition unit (model vector) number

M = input vector group number

MOD(L,I) = ith digit of lth model vector